Development of a DSP-32C Based Electrocardiograph Machine using Microcomputers

by

Wasim Tawfik Sulaiman Faddah

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

October, 1992

Order Number 1354070

Development of a DSP-32C based electrocardiograph machine using microcomputers

Faddah, Wasim Tawfik Sulaiman, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

DEVELOPMENT OF A DSP-32C BASED ELECTROCARDIOGRAPH MACHINE USING MICROCOMPUTERS

BY

WASIM TAWFIK SULAIMAN FADDAH

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

OCTOBER, 1992

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN, SAUDI ARABIA
COLLEGE OF GRADUATE STUDIES

This thesis, written by ***WASIM TAWFIK SULAIMAN FADDAH*** under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of ***MASTER OF SCIENCE*** in ***ELECTRICAL ENGINEERING***.

Thesis Committee

Dr. M.S.El-Hennawey (Advisor)

Dr. A. Al-Khadra (Member)

Dr. A.K. Al-Ali (Member)

Dr. S.S. Penbeci (Member)

Department Chairman

Dean, College of Graduate Studies

TO ALL WHOM I LOVE

To the stone rising up high in the sky

To the child throwing the stone

To the country holding the child

TO : **PALESTINE**

To the soil I had my first playground on

TO : **SYRIA**

To the land gave me this chance

TO : **SAUDI ARABIA**

To the candles burning giving light to my life

TO : **MY MOTHER AND FATHER**

To the nice and wit person I've grown with

TO : **MY SISTER**

To the only person I feel as my image

TO : **MY BROTHER**

ACKNOWLEDGEMENTS

First of all, praise be to ***ALLAH*** whose help made it possible to complete this research.

Acknowledgment is due to King Fahd University of Petroleum and Minerals for the support of this research.

I wish to express my sincere gratitude to my thesis advisor, **Dr. M. Samy El-Hennawey** , for his patient guidance and his generous support during the course of this work. I would like to express my deep appreciation to my committee members, **Dr. Akram Al-Khadra** from King Faisal University, **Dr. Abdul-Rahman K. Al-Ali** and **Dr. Suleyman Penbeci** , for their valuable suggestions, helpful remarks and kind cooperation.

Thanks to all my friends and colleagues, especially **Mr. Homam A. Hashim** and **Mr. Saleh A. Al-Meshari** for their support and help to complete this work.

Finally, special thanks to my parents, sister and brother for their encouragement, inspiration and unselfish support during my studies.

خلاصة الرسالة

اسم الطالب الكامل : وسيم توفيق سليمان فضة

عنوان الدراســـة : تطوير مخطط بياني كهربائي لعمل القلب باستخدام معالجة الاشارات الرقمية على أجهزة الميكروحاسبات

التخـــصـــص : هندسة كهربائية

تاريخ الشهـاده : جمادي الأولى ١٤١٣هـ / تشرين الأول ١٩٩٢م

تقوم أجهزة التخطيط البياني الكهربائي لعمل القلب بدور هام في مساعدة الأطباء لإنقاذ حياة البشر .

لقد تم تطوير فكرة دمج أجهزة الحاسب الآلي الشخصية ولوحة معالجة الإشارات الرقمية في تحصيل وتحليل وعرض إشارات أجهزة التخطيط البياني الكهربائي لعمل القلب في هذه الأطروحة . ويهتم الجزء الرئيسي في هذه الأطروحة بتحصيل المعلومات من موصلات جهاز التخطيط الكهربائي القلبي . ولقد تتطلب البحث تطوير وتركيب أجهزة ربط ومعالجة الإشارات الكهربائية القلبية . وتضمن العمل ايضا تطوير برامج حاسوبية تشتمل على لغات عاليا مثل لغة " C " وخفيضة مثل لغة " التجميع " . وقد تم ذلك بإستخدام لوحة معالجة الإشارات الرقمية والمسماه DSP-32C . وكذل فقد تمت دراسة القلب كعضو حيوي فيزيائي . بالإضافة الى ذلك فقد أجريت دراسة الطرق المستخدمة في تحليل الإشارات الكهربائية القلبية . وقد تم التحري عن إمكانية إستخدام وتطبيق هذه الطرق لتحصيل قيم الموجات المتضمنة في إشارة القلب الكهربائية . وقد قمنا بدراسة كاملة عن لوحة معالجة الإشارات الرقمية DSP-32C ومعالجها الرئيسي DSP32C . وأخيراً فقد أستخدمت لغة " C " في الحاسب الشخصي لتجميع البيانات المدخلة من مستخدم الجهاز كالمعلومات الشخصية عن المريض وكذلك لعرض الإشارات القلبية والنتائج التي حصلنا عليها من لوحة DSP-32C بإستخدام لغة " C " متكاملة مع لغة التجميع الخاصة بلك اللوحة .

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران ، المملكة العربية السعودية

جمادي الأولى ١٤١٣هـ / تشرين الأول ١٩٩٢م

THESIS ABSTRACT

FULL NAME OF STUDENT	:	WASIM TAWFIK FADDAH
TITLE OF STUDY	:	DEVELOPMENT OF A DSP-32C BASED ELECTRO-CARDIOGRAPH MACHINE USING MICROCOMPUTERS
MAJOR FIELD	:	ELECTRICAL ENGINEERING
DATE OF DEGREE	:	OCTOBER, 1992

In this thesis, the idea of integrating personal computers (PC's) and digital signal processing boards (DSP) to acquire and analyze the electrocardiographic (ECG) signals is achieved. The main part of the thesis is concerned with acquiring data from the ECG leads. A hardware design which consists of the signal interfacing and conditioning is developed and implemented. Moreover, the hardware design is integrated with a software package that uses both high and low level languages for the acquisition and analysis of the signal. This is performed by using a digital signal processing board called DSP-32C. A study about the heart as a physiological organ is performed too. Moreover, a survey of some methods analyzing the ECG signal and extracting its features is done. An investigation of applying these methods to our study is performed in order to get the values of QRS, PT, QT waves and the heart rate. In addition, a complete study of how the digital signal processing (DSP) board DSP-32C and its main CPU DSP32C chip is performed. Integrated with Assembly language of the DSP-32C board, C-language is used as the main programming language on this board in order to acquire and analyze the ECG signals.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

OCTOBER, 1992

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1 Background in Biomedical Engineering

Biomedical engineering is the branch of engineering that is concerned with the studies of medical technology. This branch of engineering is the one that helps medical doctors in the diagnosis of illness or any abnormalities in the human body with the aid of medical instruments. Biomedical engineering could be seen as the half way between medicine and electrical engineering [1].

Heart can be considered as the most vital organ in the body. Electrocardiogram (ECG) machines were invented and developed in order to be able to detect problems that may occur to the heart [2]. The ECG machines read and measure signals detected from heart by transducers called leads. These signals are then transferred into readable signals on a monitor or a paper. The healthy heart has an ideal

output signal which can be referred to as a reference in diagnosis. This ideal output looks like pulses oscillating with a given frequency (60-100 p/m for adults). Roughly speaking, any output from an ECG machine which deviates from the ideal situation, could be said to be an output from a sick heart [2,3]. Figure 1.1 shows an example of the output of an ECG machine.

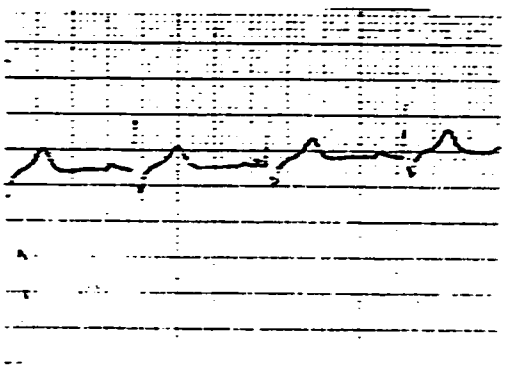


Figure 1.1: AN OUTPUT OF AN ECG MACHINE.

Each single pulse of the output consists of five parts that an M.D. would be interested in, (see Figure 1.2). These parts are P,Q,R,S and T. The ECG machine will provide, in addition to the graph output, more information about the heartbeat rate which can be seen as rate of the P wave and QRS wave. Other features are important too,

eg. P wave width and height, PR segment, QRS duration, and QT segment. [2].

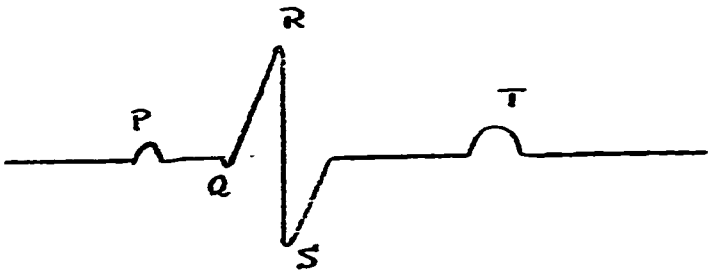


Figure 1.2: A SINGLE PULSE FROM THE HEART.

In the following section, the motivation of the thesis is considered. Section 3 will be devoted to explain the strategy of the thesis. Section 4 will propose steps of work in the thesis. Expected advantages of the thesis are shown in section 5. Finally, the thesis organization is described in section 6 of this chapter.

1.2 Motivation of the Thesis

Based on a survey on the ECG machines in the Kingdom of Saudi Arabia, it is found that high performance ECG

units are very expensive pieces of equipment. This is because every manufacturer of such a machine tries to keep the secrets of the invention as confidential as possible. Moreover, if the ECG unit is not measuring any heart signals, it has no other use. This leads to the idea of having a multi-purpose, yet efficient, inexpensive ECG piece of equipment. Personal computers (PC's) have been available with a very large volume recently. They are very versatile, and very compatible with each other. Off-the-shelf hardware and software packages can be obtained and used on PC's due to their compatibility. For example, digital signal processing (DSP) boards which can be used to process a given signal and show the main features of it can be connected to a PC [3,4]. Due to their versatility and compatibility PC's are to be used in the thesis to play the role of an ECG unit.

1.3 Main Idea of Thesis

The main idea of the thesis is to use PC's playing the role of a smart ECG equipment. This can be achieved by building an interface between the transducers used in detecting the heart signals and a PC (see Figure 1.3). The heart's pacemaker produces electromotive forces that generate magnetic fields each heartbeat. These fields are

converted into electrical signals by the transducers which, in turn, transmit this electrical signal to the interface unit.

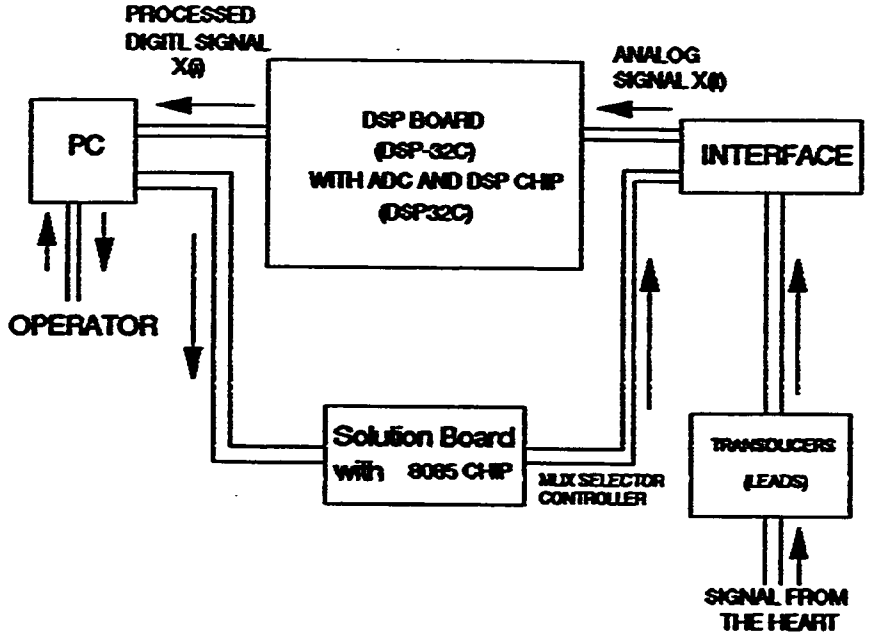


Figure 1.3: System Hardware Architicture

Two types of transducers (leads) are used in ECG units, namely, bipolar and unipolar. Bipolar leads extracts their readings by measuring the difference between two similar potentials. There are three bipolar leads that are connected to the left hand, right hand and left foot. Three measurements are read by the bipolar leads(Figure 1.4)

1. "I" between the right hand and left hand
2. "II" between the right hand and left foot

3. "III" between the left hand and left foot

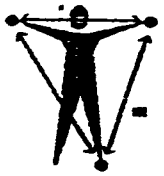


Figure 1.4: Bipolar Leads

Unipolar leads are divided into two parts, peripheral and precordial. The unipolar limb leads (peripheral) are combination of two leads that measure the difference between a potential and a relative ground (0). Three measurements are read by the peripheral leads(Figure 1.5)

1. "aVR" between the right hand and relative ground "0"
2. "aVL" between the left hand and relative ground "0"
3. "aVF" between the left foot and relative ground "0"

Six leads compose the unipolar chest leads (precordial). They are placed successively upon six points of the precordial region as shown. they are referred to by V1-V6

Figure 1.5: Unipolar Leads

The following argument explains why it is necessary to use all these leads in an ECG unit. As long as the heart is an elliptical body, and because it is divided into four parts, and any part could be responsible for a weakness, therefore, we should study heart electrical activities from all possible windows. These studies depend on the heart potential which can be divided into parts on an electrical axis. Each lead will give a different potential and sign depending on its location around the electrical axis of the heart. Figure 1.6 shows the outputs of leads I, II and III according to their replacement on the electrical axis (to be explained later in chapter 2) of the heart.

The readings of these measurements determine the status of the heart. One of the objectives of this thesis is

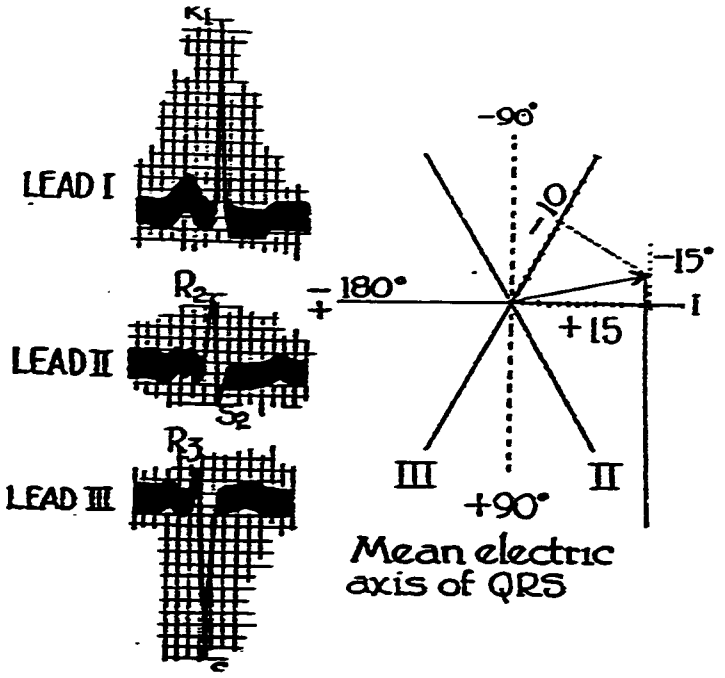


Figure 1.6: Leads Reading according to their position in the heart axis

to graphically display these readings separately or together on a screen of a PC, together with some quantities measured by using the DSP board. These quantities help the M.D. diagnose the abnormalities.

The interface plays the role of a gate that allows the transducers' analog signals to pass into an analog to digital (A/D) converter. The purpose of converting the signal into digital is to enable the DSP board to process the signal. The DSP unit is to extract the features desired in the signal such as segments intervals, variations and peak to peak values and send them to the PC.

Moreover, displaying the waveform of the heartbeat in real time will be provided. In addition, a study about the mechanism of the heart will be presented as a part of this thesis.

1.4 Work Procedure

The following course of action is taken as the procedure in the thesis

1. Some information about the diseases that can be detected by ECG are discussed. For example, the abnormality of the left bundle branch block (LBBB) could be detected from the following features:

   a. Wide QRS ≥0.12*s*

   b. V_6 *has a rsR'* shape;

   c. The T wave is headed downwards in leads I and V_6

   d. the loss of R wave in v_1.

   While the right bundle branch block (RBBB) abnormality could be detected from the following characteristics in the ECG readings:

a. Wide QRS ≥0.12*s*

b. Reading V_1 *has a tall R wave with a rsR'* shape ;

c. The T wave is headed downwards in leads V_1

d. The S wave is wide and deep in readings I and V_6.

These information could be found in chapter 2.

2. A study on the system hardware such as DSP boards and interfacing units is performed. Moreover, the software of the PC and the DSP board (assembly and C languages) is developed. The software is presented in chapter 3 where the digital signal processing board and chip are studied.

3. Design a proper interface between the transducers and the A/D converters. This is needed because the original signal that comes from the heart is very weak and may be corrupted. The interface unit amplify and cleans the signal.

4. Assemble the system.

5. Perform testing on the assembled system. Explanation of steps 3,4 and 5 can be found in chapter 3

6. Investigation of the possibility of smart processing ECG signals to display automatically the quantities an M.D. may be interested in measuring.

1.5 Advantages Sought by Thesis

As already discussed the objective of this project is to produce a smart ECG unit based on microcomputers and digital signal processing (DSP) boards. The versatility of the microcomputers and software in general allows to design and implement an integrated piece of equipment that opens the door for the possibility of interfacing more than one biomedical instrument (eg ECG, electro-encephalogram (EEG).....etc) to one microcomputer. Receiving different signals from different types of transducers will enable us to use microcomputers as multi-purpose biomedical equipments in the future.

1.6 Thesis Organization

In this chapter an introduction to a microcomputer DSP based electrocardiogram ECG design is reviewed. In Chapter 2, a summary about the heart and its electrical activity is presented. In Chapter 3, a survey about the use of microcomputers and signal processing in ECG is presented. Chapter 4 summarizes the work performed and displays the results obtained in the laboratory tests. Moreover, conclusions and recommended future work are presented in Chapter 5. Finally, appendices A,B and C list software used in our work, while appendices D and E shows PCW and the IOC registers tables respectively.

1.7 Summary

In this chapter an introduction to the design of a microcomputer DSP based electrocardiogram ECG is presented. In the next chapter, a summery about the heart is displayed. It describes the function and anatomy of the heart, the cardiac cells and electrical activity of the heart and its cells.

Chapter 2
HEART

2.1 Overview

In the previous chapter an introduction to biomedical engineering is presented. Moreover, an overview of the thesis and the work procedure is introduced, and the literature review is discussed. In this chapter, the heart, as a vital organ for the body is presented as follows. We start with the anatomy and function of heart. Then the electrical activity of myocardial cells is presented. Moreover, the ventricular cell behavior is illustrated. In addition, the body surface potentials is depicted. Last, the cardiac vector with an illustration of bipolar and unipolar leads of an ECG is discussed.

2.2 Anatomy and function of the Heart

The heart is a hollow elliptical organ, suited in the thorax between the lungs ,above the diaphragm and behind the sternum. In adults its average size is 4.7" in length, 3.5" in width and 2.3" in thickness. Its left

ventricle (LV) mass is $87g/m^2$ in male adults, and $69g/m^2$ in female adults. The use of this standard measure of mass has resulted because the LV is the dominant part of the heart. Its shape roughly is of an inverted cone that its base is directed somehow upward, backward and to the right, and the pointed point or apex is directed somehow downward, forward and to the left. It is rotated so that the right side is almost in front of the left side [5].

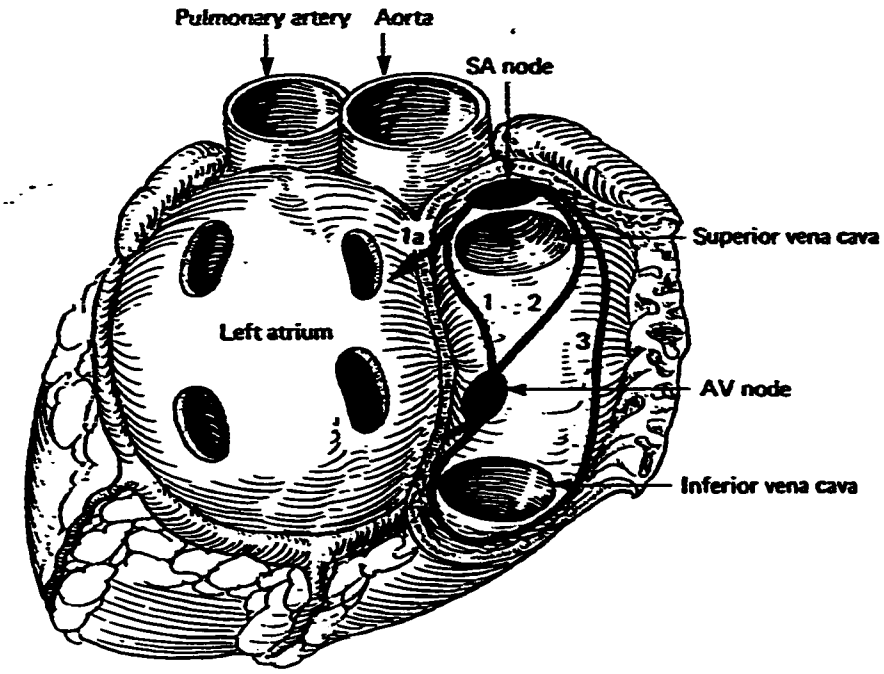


Figure 2.1: heart.

The heart has four chambers, the left atrium and ventricle, and the right atrium and ventricle [3]. The atrial function is to receive the blood from the venous system and direct it to the corresponding ventricle. The atrium contracts at the end of the ventricular diastole to augment filling of the ventricle. This is very important in the cases of poor compliance of the ventricle. However, 70% of the filling of the ventricle is passive, while the atria are merely not resistive to the blood during ventricular diastole.

The ventricles, on the opposite side, have to pump the blood into both the systemic and pulmonary circulations. The peripheral resistance of systemic circulation is higher than that of the pulmonary one. Thus the left ventricle (which pumps the blood into the systemic circulation) has thicker walls than the right ventricle that pumps the blood into the pulmonary circulation.

As mentioned earlier the heart serves as a four-chambered pump for the circulatory system (see Figure 2.2). The main pumping function is carried out by the ventricles, while the atria are merely antechambers to store blood during the time the ventricles are pumping. The resting or filling phase of the ventricles is

referred to as ***"diastole"***. The contractile or pumping phase is called ***"systole"*** [2,3,5].

The heart has an intrinsic electrical activity that controls the mechanical function of the heart. This in turn is under the influence of the nervous system [6].

The specialized conduction system constitutes only a small portion of the heart. The walls of the left ventricle is 2.5-3.0 times as thick as the walls of the right ventricle (this is because of the nature of the task each of them is involved in), while the interventricular septum is nearly as thick as the left ventricular wall. The major portion of the muscle mass of the ventricle consists of the free walls of the right and left ventricles and the septum. Considering the heart as a bioelectric source, the strength of this source can be expected to be directly related to the mass of the active muscle(i.e., the number of the active myocytes or myocardial cells). Therefore, the atria and the free walls of the septum of the ventricles can be considered the major contributors to external potential fields from the heart.

The rhythmic cardiac impulse is originated in pacemaking cells in the ***sinoatrial (SA)*** node, located at junction of superior vena cava and right atrium. The pulse is

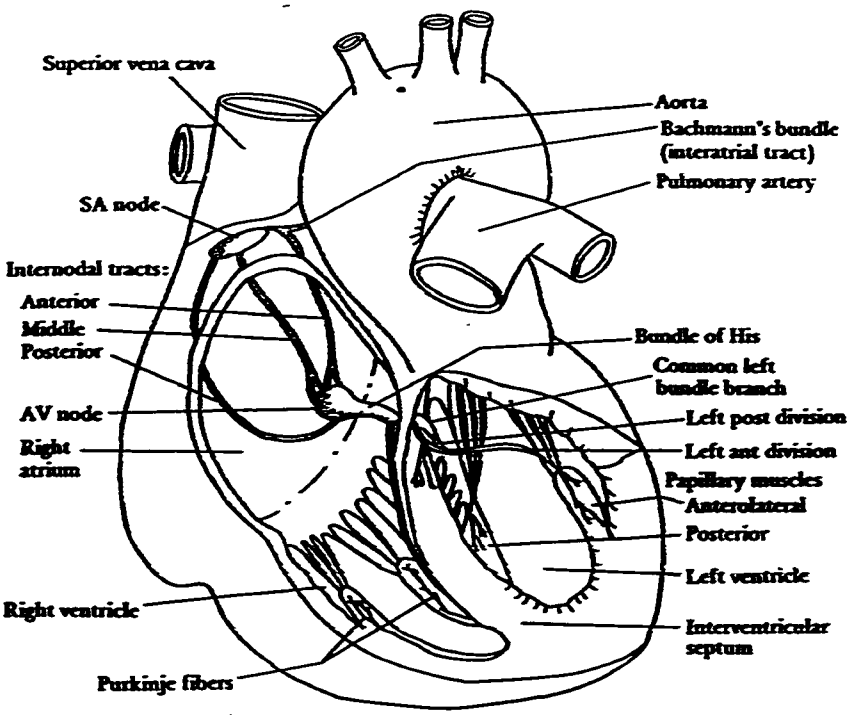


Figure 2.2: Anatomy of Heart

then travels to the ***atrioventricular (AV)*** node. The impulse is then delayed at the AV node, before it continues into the Bundle of His, the right bundle branch, the common left bundle branch, the anterior and posterior divisions of the left bundle branch and ***Purkinji network.*** Right bundle branch runs along right side of interventricular septum to apex of right ventricle before it gives off significant branches. Left common bundle crosses to left side of septum and splits into anterior and posterior divisions. [2].

2.3 Electrical Behavior of Cardiac Cells

The heart is comprised of several different types of tissues(SA and AV nodal tissue; atrial, Purkinji, and ventricular tissue). Representative cells of each type of tissue differ anatomically to a considerable degree. They are all electrically excitable, with each type of cell exhibiting its own characteristic action potential. [2].

2.4 The Ventricular Cells

The ventricular myocardium is composed of millions of individual cardiac cells. Each cell is 15*15*150 ηm in volume [2]. A cell is shown in figure 2.3 under a light microscopy. As mentioned earlier, all cells are electrically excitable. Although cells are relatively long and thin which makes them run generally parallel to each other, there is a considerable branching and interconnecting between them ***(anastamosing).*** Plasma membrane surrounds the cells which makes an end-to-end contacts with adjacent cells at a dense structure known as ***intercalated disk.*** Each fiber contain many contractile myofibrils that follow the axis of the cell at a dense structure called ***the myofibrils,*** that follow the axis from one end (intercalated disc) to the other.

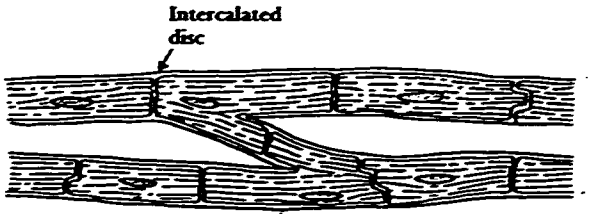


Figure 2.3: A Cardiac Cell

These myofibrils are responsible for the ***contractile machinery*** of the heart. The cardiac cells in the cardiac tissue are in intimate contact at the intercalated discs, both electrically and mechanically, which makes the heart muscle works as a unit.

Prior to excitation, the typical ventricular cell has a resting potential of about "-90 V". The initial rapid ***depolarization*** which is the propagation for excitation which has the trigger for releasing the stored energy in the cardiac muscle has a rate of rise usually greater than 450V/s [7]. After this phase, an initial rapid phase of repolarization leads to a maintained depolarization plateau region that lasts approximately for 200-300 ms. A final repolarization phase follows which restores the membrane potential to the resting level (-90V) and is

maintained for the remainder of the cardiac cycle. The duration of the action potential waveform is referred to by the term ***electrical systole*** while the term ***electrical diastole*** refers to the resting phase. The addition of the electrical action of the ventricular cells would sum up to give a specific shape of the heart electrical cycle.

2.5 Ventricular Activation

A great deal of studies were performed on ventricular activation. Most of the information are given by studies conducted on experimental animals. However, Figure 2.4 is a map of ***isochronous (synchronously excited)*** excitation surfaces for a perfused heart of a human being who died from a noncardiac condition. The isochronous excitation surfaces is mapped through the study of the time of the time of arrival of electrical activation. It is noted that electrical activation starts at the septal surface of the left ventricle (5 ms into the QRS complex) and then the activity spreads with increasing time in the direction from left to right across the septum. As time passes excitation spreads and tends to be more confluent. For example, at 30 ms nearly closed activation surface is seen. Excitation then proceeds in an ***epicardial*** (outside the heart) direction.

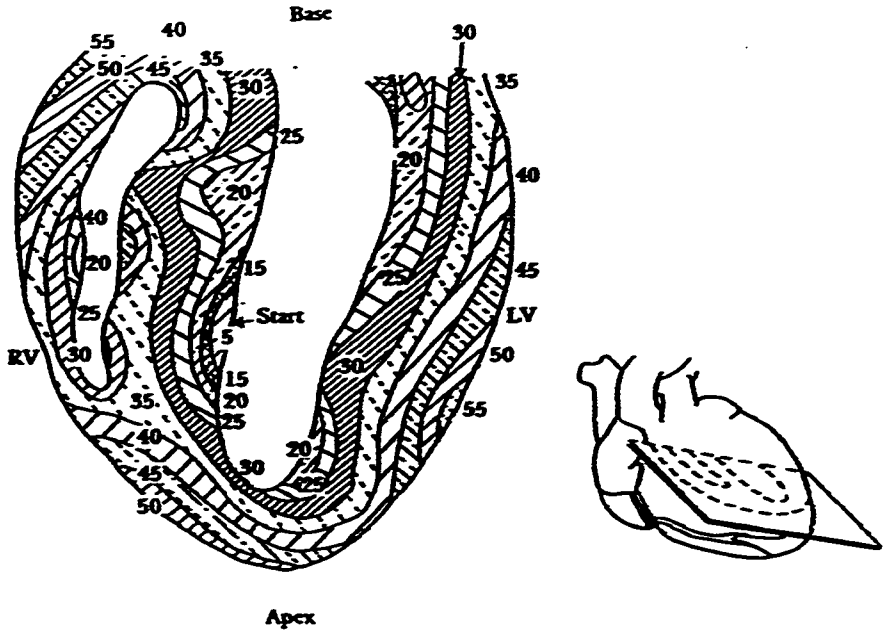


Figure 2.4: Isochronous Excitation

The isochronous electromotive surface propagates through the myocardium in an outward direction from the ***endocardium*** (inside of the heart). Due to high interface between the individual cardiac cells which are the seat of the electromotive force, many of them are active simultaneously in a localized area. The anatomical substrate for this electrical interaction is the high degree of branching of individual cardiac cells and the low resistance of the intercalated discs at the junctions between the cells [2]. Figure 2.5 represents the electro-

physical behavior within a small segment of the myocardial wall. The ventricular action potential is represented by $\Phi_m(z)$ where z is the direction of propagation. The waveform proceeds at a constant propagation velocity so that spatial and temporal events can be interchanged, because potential must be of the form f(z-ut), where u is the conduction velocity. This leads to a difference of potentials $\Delta\Phi$ between measurements taken by a sequential pair of closely spaced electrodes oriented in the direction of propagation of the wavefront. $\Delta\Phi$ is given by

$$\Delta\Phi = \frac{d\Phi_m(z)}{dz}\Delta z \qquad (2.1)$$

where Δz is a small electrode spacing < 1 mm [2].

This will make $\Delta\Phi$ proportional to $\frac{d\Phi(z)}{dz}$ which is approximately an error function [2], *i.e.*,

$$\frac{d\Phi(z)}{dz} = erfc(z). \qquad (2.2)$$

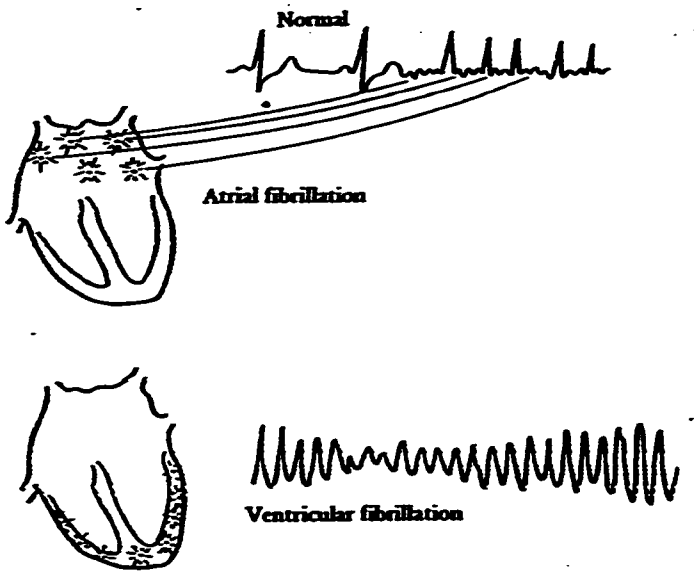


Figure 2.5: Electrophysical Behavior

2.6 Body Surface Potentials

Because of the epicardial direction of the electrical activation (excitation) of the ventricle, a closed-line action currents are produced in the thoracic volume conductor (considered to be a purely passive or conductive medium containing no electrical sources or sinks). Potentials measured at the outer surface of this medium ,i.e., on the body surface, are referred to as ***electrocardiograms*** or ***ECGs***

In the electrocardiographic studies, the heart is viewed as an electrical equivalent generator. A common assumption is that, at each instant of time in the sequence of ventricular activation, the electrical activity of the heart can be represented by a net equivalent current dipole located at a point that is called ***electrical center*** of the heart. This center is assumed to be within the anatomical boundaries of the heart [4].

In the case where several regions of both ventricles are simultaneously active (which might be the case), the electrical activity of each region at any instant of time could be thought of as being a current dipole and net dipolar contribution from all active areas determined at the electrical center. The thoracic medium could be thought as the resistance load of this equivalent cardiac generator. Because of this difference, potentials between different points on the body could be measured.

The electrocardiographic output measured from electrodes on the surface of the body is a measure of magnitude of a body-surface-potential plotted versus time. The main significant features of the waveform are the P, Q, R, S and T waves, the duration of each wave, and certain time intervals such as PR, ST, and QT intervals.

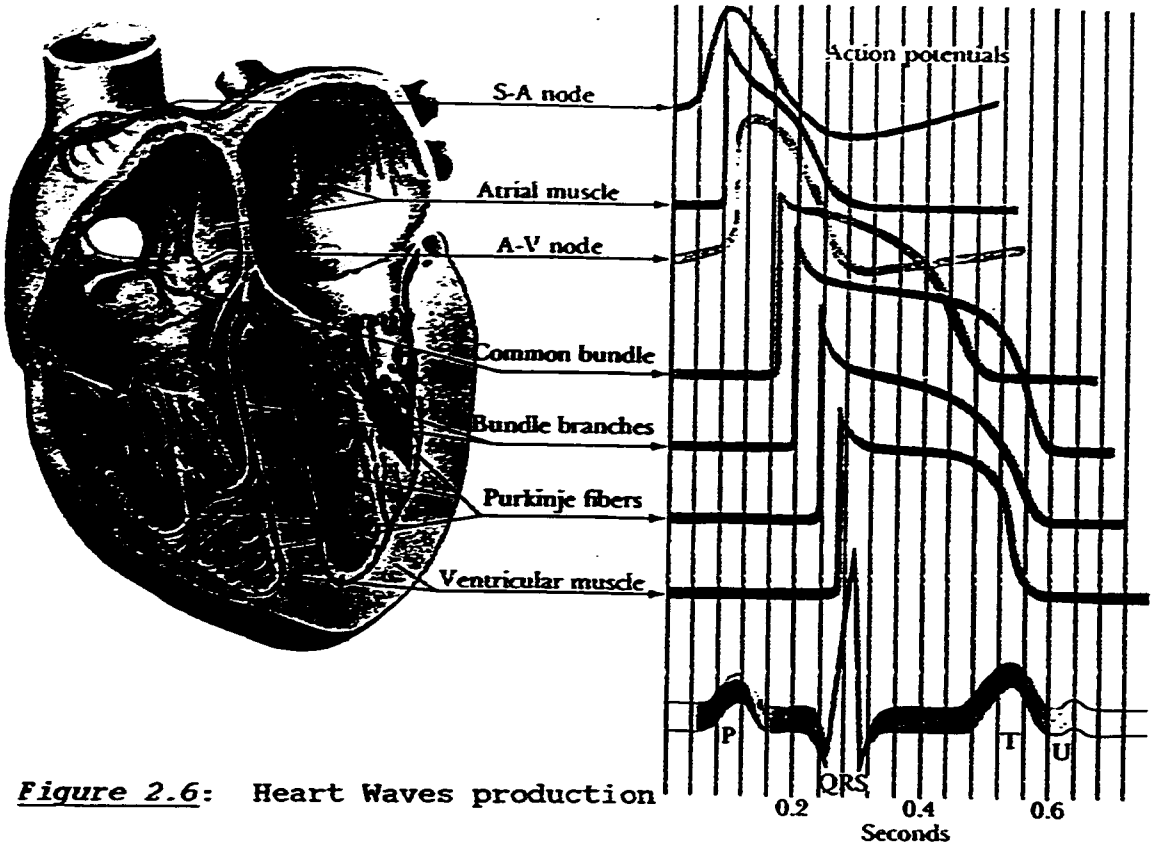


Figure 2.6: Heart Waves production

From figure 2.6 it is clear that the P wave is produced by the atrial depolarization, the QRS complex by ventricular depolarization primarily and the T wave by ventricular repolarization. The manifestations of atrial repolarization are normally masked by the QRS complex. The PR and ST segments are normally at 0 potential, with the PR interval being caused mainly by conduction delay in the AV node. The ST segment is related to the average duration of the plateau regions of individual ventricular cells. A small additional wave called the U wave is

inconsistently recorded after the T wave. It is believed to be due to the slow repolarization of ventricular papillary muscle.

As a final remark on excitation, it must be noted that situations are commonly encountered in which there is an uncoupling of the excitation mechanism, and excitation can occur without contraction. In addition, a premature ventricular excitation may not be followed by a mechanical contraction strong enough to open the aortic or palmonic valves. This produces a number of R waves greater than the number of atrial pulses [7].

2.7 Cardiac Vector

As mentioned earlier, the term "electrocardiogram" is specifically reserved for the record of electrical activity of the heart obtained with the body-surface electrodes. These electrodes should be placed in a specific areas of the body in order to get maximum benefit of the readings of an ECG. This placement is done according to a ***cardiac vector*** that characterizes the heart. The term cardiac vector designates all of the electromotive forces of the heart cycle. It must be appreciated that any given instant during depolarization or repolarization, electri-

cal potentials are being propagated in many directions in the space. Over 80% of these potentials are cancelled out by opposing forces, and only the net is recorded.

The ***instantaneous vector*** represents the net electrical forces at a given time. A ***mean vector*** of any given portion of the heart cycle (e.g.,QRS) represents the mean magnitude, direction, and polarity for that period. VCG is or spatial vectorcardiogram is available to measure the vector for atrial depolarization (QRS complex) and ventricular repolarization (T wave).

Einthoven devised a vector frontal plane of the heart depending on the following assumptions (Figure 2.7):

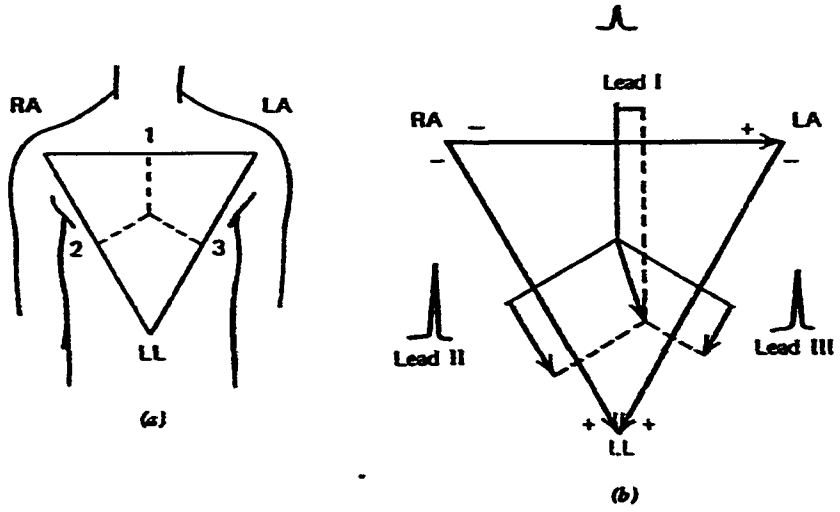


Figure 2.7: Einthoven Cardiac vector

1. The dipolar hypothesis which states that each instantaneous vector is the result of an electrical potential which acts as a dipole.
2. The center of electrical activity of the heart is located in the anatomic center of the chest.
3. Leads I, II, and III, which are measurements to be described later, are equidistant from the center of electrical activity.
4. The human torso can be assumed to be of a spherical shape.
5. All body tissues and fluids can be assumed to be equally good conductors of electrical potentials.

The standard leads in the ECG machine are those which might satisfy the second assumption of Einthoven cardiac vectors. These are I, II and III. The readings of these leads are read by measuring the potential difference between the left arm (LA), right arm (RA) and left leg (LL) electrodes. The assumption is that these leads (i.e., I, II, and III) constitutes the three legs of Einthoven triangle [3]. They can be calculated easily by measuring the difference in potential between the three electrodes LA, RA, and LL.

$$I = LA - RA \tag{2.3}$$

$$II = LL - RA \tag{2.4}$$

and $III = LL - LA$ (2.5)

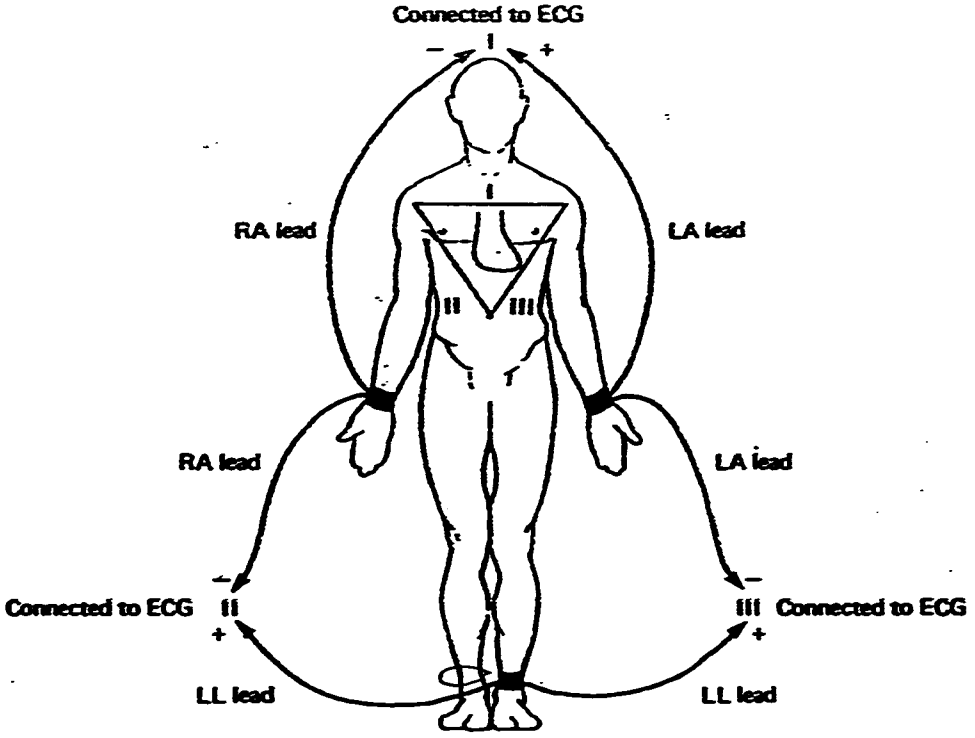


Figure 2.8: Standard leads connections

2.8 Bipolar Leads

Depending on Einthoven triangle, KVL, KCL, assuming that the body is a closed circuit and noticing that the polarity of lead II is of an inverted nature, we can derive the following equation

$$II = I + III \quad (2.6)$$

The I, II and III leads which are the original leads selected by Einthoven are called the bipolar standard leads. Another type of bipolar leads are the chest (c) leads, which take their readings from measurements between any given position on the chest and one extremity. (figure 2.9). However, this type of leads are of no use nowadays.

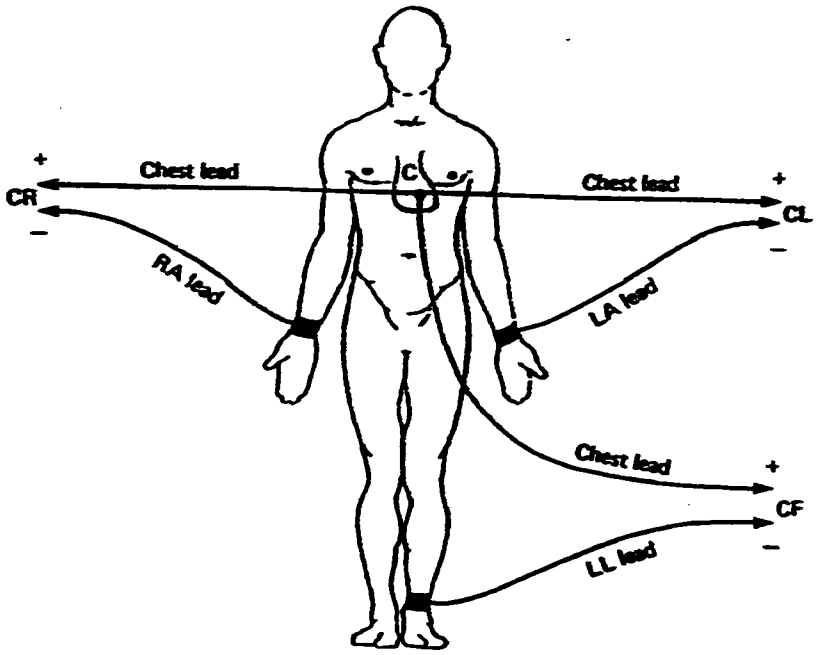


Figure 2.9: complete set of Bipolar leads

2.9 Unipolar Leads

Another important type of leads are the unipolar leads. These leads are called the vector leads. The name came from the fact that they measure the absolute cardiac potential according to the reading of one lead. For exam-

ple, VR reads the absolute reading of potential at the right arm [6]. This is not explicitly right because the readings are again between a specific lead and an indifferent lead that represents the ground.

Before introducing the indifferent ground, a lead that is connected to the right leg (RL) is discussed. This lead is used for protection purposes. This happens by connecting it to the ground and connecting all the other leads to it via resistors. This protects the ECG machine from the large currents that might be produced from the high input impedance it is driven through. This is explained in the coming chapter.

The indifferent lead serves as a ground. Its idea comes from the assumption that the body is a closed circuit which means that the connection of the three standard leads will give a net potential equals to 0. This is referred to as the T lead that is defined as [6]

$$T = RA + LA + LL = 0 \qquad (2.7)$$

This means that the difference between any X lead's potential and the indifferent lead's potential will result in a net potential that is equal to the X lead's potential. figure-11 shows the connection for the

indifferent lead. The readings from the unipolar leads are

$$VR = RA - T \tag{2.8}$$

$$VL = LA - T \tag{2.9}$$

$$\text{and} \quad VF = LL - T \tag{2.10}$$

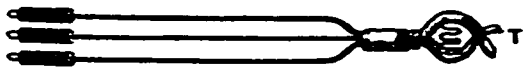


Figure 2.10: connections for the T lead

and the potentials read from these leads are called the vector potentials.

Another type of leads are discussed as an extension to the idea of the unipolar leads which produce the vector potentials, namely those leads which produce the augmented vector potentials. The augmented vector potentials produce enhanced potentials with about 50% enhancement from the vector leads. The idea is discussed as follows:

> The measurements are taken by taking the difference between any electrode and the average between the other two electrodes. As an example aVR which is the difference potential measured between the RA electrode and the average between the other two electrodes LA and LL. This will lead us to the following equations:

$$aVR = RA - \{\frac{LL+LA}{2}\} \tag{2.11}$$

knowing that

$$RA + LA + LL = 0 \tag{2.12}$$

then

$$aVR = RA + RA/2 = 3/2RA = 3/2VR. \tag{2.13}$$

which gives an enhancement of 50 % from the vector leads [6].

2.10 Summary

In this chapter an overview about the heart and its function is displayed. Moreover, the electrical behavior of the heart cells is presented. The body surface potential and the cardiac vector of the heart is discussed too. Last, an illustration bipolar and unipolar leads of an ECG is discussed. In the next chapter a display of the of the usage of microcomputers in ECG is to be explained. Moreover, the digital processing literature review is presented too.

Chapter 3
SIGNAL PROCESSING AND MICROCOMPUTERS IN CARDIOLOGY

3.1 Overview

In the previous chapter the heart as a physiological organ is presented. In this chapter, digital processing techniques are presented and a survey on the uses of microcomputers in cardiology is displayed.

In the following section some algorithms for the detection of QRS complex in the ECG signal are illustrated. They use different approaches for this purpose. Moreover, in the section after, some of the attempts in using microcomputers in cardiology are described.

3.2 Signal Processing of the ECG signal

Here we review some digital processing techniques of the ECG signal to incorporate them to our smart ECG machine. Four QRS detection algorithms are described. These algorithms are based on amplitude and 1st derivative, 1st and 2nd derivative, digital filters and linear prediction scheme [8].

The electrocardiogram (ECG) is an important tool for providing information about the heart. Its main output is an electrocardiograph that shows the heart's signal which is formatted due to polarization and depolerization of the heart (refer to Figure 1.2). The ECG signal contains many waves. The most important wave is the QRS. The attention is given to the detection of the QRS wave in recent research [2,3,7].

Many algorithms have been developed for the detection of QRS signal depending on various bases. In this presentation, the aforementioned four algorithms are discussed. Next section is devoted to the discussion of a detection algorithm based on amplitude of ECG signal and 1st derivative. Then an algorithm depending on 1st and 2nd derivative will be presented. Later, an algorithm depending on digital filters is illustrated. Finally, the linear prediction based method for QRS detection is depicted.

The selection of these algorithms is made due to a comparison of many detection algorithms that were presented in reference [8] three of them were the best, and the linear prediction based algorithm is a very good tool for later investigation of feature extraction of the QRS signal.

3.2.1 Remarks about the ECG signal used in the algorithms to be Mentioned

The ECG signal in reference [8] was like a gold standard recorded signal and digitized manually. The original ECG recording contained 37 cycles with a total time of 32s. These cycles were sampled into 8192 samples. This means that sampling frequency = 256 samples/s.

All previous algorithms are used for QRS detection in the presence of noise.

3.2.2 Algorithm Based on ECG Signal Amplitude and 1st Derivative

Given that the shape of the ECG has a lot of differences in magnitude due to the presence of different voltages that compose the different waves of the ECG, an algorithm that depends on the ECG signal amplitudes for

feature extraction is created. A block diagram of the algorithm developed by Fraden and Neuman [9] is shown in Figure 3.1. The algorithm is described analytically as follows:

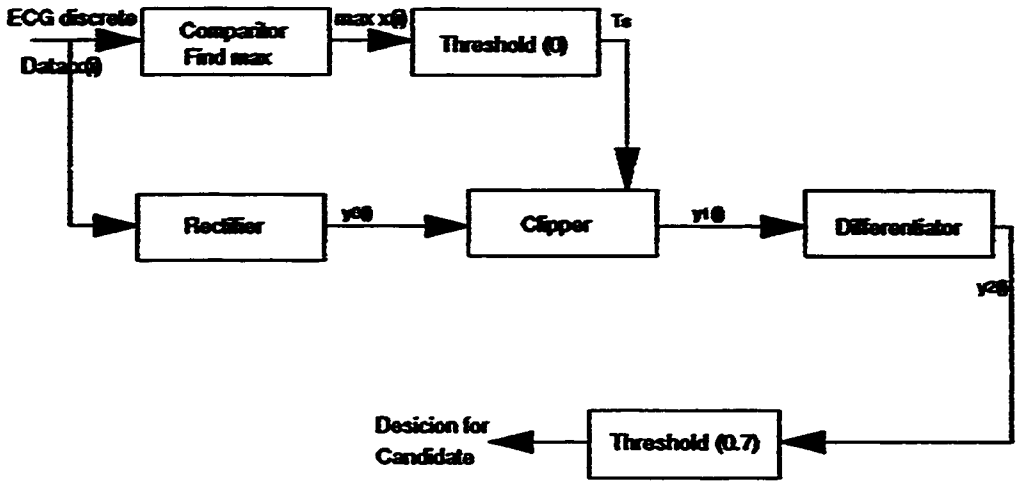


Figure 3.1: Amplitude and 1st derivative QRS detection algorithm block diagram

1. Find max(x(n)), where x(n) is the digitized ECG signal.
2. Construct an amplitude threshold Ts= 0.4 max(x(n))
3. Absolute value of the raw data is found

$$Y0(n) = \begin{cases} x(n), & x(n) \geq 0 \\ -x(n), & x(n) < 0 \end{cases} \qquad 0<n<8191 \quad (3.1)$$

4. Rectified data is passed through a clipper

$$Y1(n)=\begin{cases} Y0(n), & Y0(n)\geq Ts \\ Ts, & Y0(n)<Ts \end{cases} \qquad 0<n<8191 \quad (3.2)$$

5. The 1st derivative is calculated by

$$Y2(n)=Y1(n+1) - Y1(n-1) \qquad 1<n<8191 \quad (3.3)$$

A QRS onset candidate is chosen when a point in Y2(n) exceeds a fixed threshold Y2(n) > constant threshold depending on the values of the ECG signal data.

3.2.3 Algorithm Based on 1st and 2nd Derivative

Another algorithm is developed by Ahlstorm and Tompkins [10]. This algorithm is built depending on the fact that the rectified derivatives of the ECG signal will result in getting finally the QRS wave solely according to its large amplitude relatively with the other waves in the signal.

3.2.4 Algorithm Based on Digital Filters

Because the QRS complex has relatively high frequency components with the other waves, many algorithms are devised based on digital filtering of the ECG signal. One of these approaches is given by Okada [11].

3.2.5 Algorithm Based on Linear Prediction

Linear prediction has been a powerful tool in digital signal processing. It's importance in ECG signal processing rises from the information that could be extracted from the residual error signal(RES). As a preliminary explanation of linear prediction the signal could be estimated from past samples. The difference between the current sample and its estimate forms the residual error [12].

Several significant advantages are found when an ECG signal is processed with linear prediction. First, the prediction order is not needed to be more than 2 because the sampling rate is usually less than 1000 Hz [13]. Second, RES contains a lot of information about the ECG signal. Third, the relative easiness in computations using linear prediction. Fourth, the accuracy in estimation of each signal parameters. Last, the high speed of computation.

Assume that the real ECG signal is s(i) and the estimated signal is

$$\hat{s}(i) = \sum_{k=1}^{p} a(k)\ s(i-k) \qquad (3.4)$$

then, the residual error signal (RES) is

$$e(i)=s(i)-\hat{s}(i)$$

$$=s(i)-\sum_{k=1}^{p} a(k)\ s(i-k) \qquad (3.5)$$

and the error energy is obtained as

$$Err=\sum_{i=1}^{N-1}|e(k)|^2 \qquad (3.6)$$

In order to be able to process the ECG signal the block diagram shown in Figure 3.2 is performed. This algorithm is presented analytically by the following:

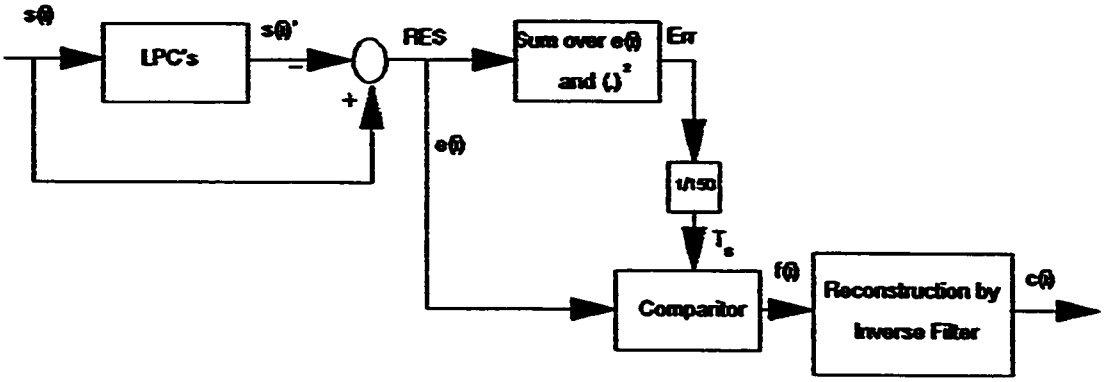


Figure 3.2: Linear prediction based QRS detection algorithm block diagram

1. Construct a threshold Ts =Err/150

2. Construct a clipper that acts as follows:

$$f(i)=\begin{cases}0, & \left|\frac{e(i)}{e(n)}\right| \le Ts \\ e(i), & \left|\frac{e(i)}{e(n)}\right| > Ts\end{cases} \qquad (3.7)$$

3. Reconstruct c(i) which contains most information of QRS wave from original set of LPC's and the center-clipped signal f(i). This is accomplished by the reconstruction of the inverse filter of the linear prediction model.

In ECG linear prediction the system model is an autoregrissive (AR) model because of the two following reasons:

1. It is easier than the moving average (MA) or autoregrissive moving average (ARMA) models in computation, because of mathematical easiness dealing with poles rather than dealing with zeroes [12].
2. The situation of not being able to have the input sequence of the ECG signal available in the early stages of the process which makes it difficult to choose the AR model.

The importance of the linear prediction method is that many algorithms for feature extraction of a signal depend on the LPC's of the signal. This is because the error signal contains a lot of information about the waves in the ECG signal especially the QRS complex.

3.2.6 Comparison Between Methods

As a result of a study performed by Friesen et al [8], it was found that the first three methods that we have chosen for our application give a very good performance in the presence of noise. It is possible to extend the first method for the detection of the P and T waves.

Some deficiencies in the presence of the electromyographic (EMG) and composite noise were detected in the performance of the second method, while the first method has some deficiencies in the presence of respiration noise. The third method performs in a satisfying way in the presence of all types of noise [8].

The fourth method is an attractive one because it holds a lot of good information about the QRS complex in the RES. However, the RES does not hold any information for the other waves in the ECG signal which is considered as the disadvantage of the method.

The first method which is based on ECG signal amplitude and first derivative is used in this project due to its simplicity and because we could extend it to detect the P and T waves.

3.3 Microcomputers in Cardiology

The idea of using microcomputers in medicine has arisen due to the revolution of microcomputer technology and the versatile applications they can offer.

Applications of microcomputers to medicine have been given emphasis since the evolution of PC's. In this chapter, some of the attempts to use PC's in medicine computer applications in cardiology are surveyed. Most of these applications give emphasis on the use of microcomputers in detecting cardiac infirmities, holter monitor, intensive care unit (ICU) and simulation of ECG lead signals uses. This implies that a continuous monitoring of one or two lead signals is sufficient for such applications. Moreover, the simulation of the ECG lead signals needs no interaction between a patient and the ECG machine. As mentioned later in chapter 4, our application of microcomputer in cardiology differs from the previous work in the following aspects:

1. the use of a digital signal processing board which gives a wide range of future applications in the field of analysing and processing the ECG lead signals;
2. the ability of using microcomputers as both an ICU unit or an ECG piece of equipment which implies the continuous display of one lead signal or the display of all lead signals simultaneously.

In the following, a summary of some of the research done is presented for different approaches in the application of microcomputers in cardiology.

3.3.1 Cardiac Arrhythmia Simulators

Le-Huy, Yvroud and Dion [14] proposed a versatile cardiac arrhythmia simulator whose generated output could be used in simulated testing of medical equipment.

The basic idea of the research depends on that a basic ECG signal could be successfully approximated by a series of fifteen linear segments. Figure 3.3 shows the linearized model with different waves (p, QRS, T), and different segments (PR, ST.....).

Le-Huy, Yvroud and Dion in [14] proposed that each segment be characterized by two codes: a duration code and a slope code. In the course of work they have characterized the following:

1. The time axis is divided into N equal sections with T time divisions per section. This means that an ECG waveform is measured by N sections or N*T time divisions. The temporal length of each time division is determined by

Temporal length of time division =

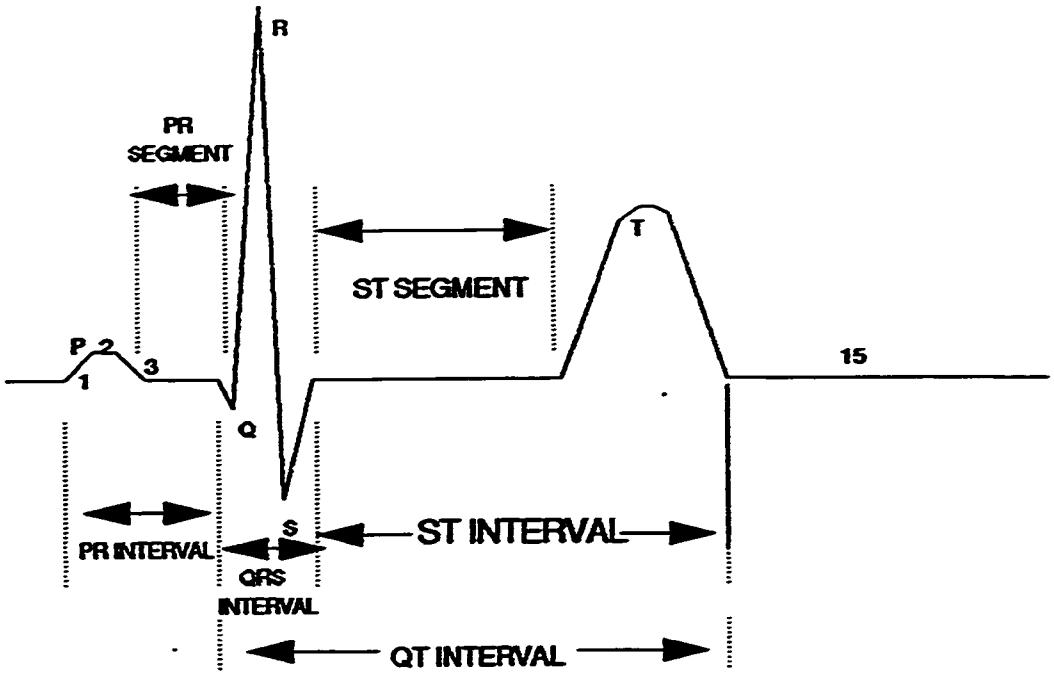


Figure 3.3: Linearized ECG Signal

$$\frac{\textit{period of ECG signal}}{N*T}$$

A simple delay loop in the main program is used to set the real-time value of this temporal length, therefore, the rate of the ECG signal could be varied at will.

2. The proportional time division is applied for each segment by considering its percentage of length with respect to the defined basic period. The duration code of each segment is then defined as:

Duration code = Number of sections * T

Its numerical value determines the number of time divisions that the main program has to loop prior to the progression of simulation.

The microcomputer section in [14] consists of an MSC800 CMOS microprocessor, an MSC810 RAM I/O Timer, 2K * 8 EPROM which contains the main program preprogrammed codes for normal derivations and some standard arrhythmias. The 128-byte RAM in MSC810 are reserved for stack operation and temporary storage. The interface with the analog output world is fulfilled by an 8-bit CMOS D/A convertor. A linearized phase low-pass filter is employed to provide a smooth ECG waveform rather than a staircase. Figure 3.4 illustrates the hardware description of the system.

The software is written so that it provides V3 lead reading by default. If a user wants to simulate another lead he/she has a an access through an interrupt driven software. During the choice, the user is capable of changing the duration and slope codes. This gives him/her the ability to choose from the leads to simulate.

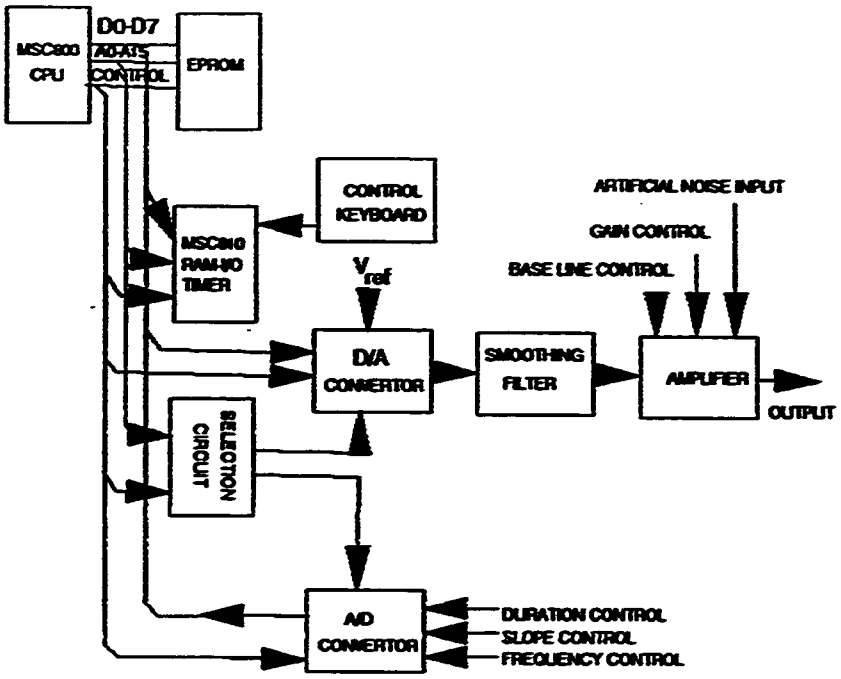


Figure 3.4: Hardware Schematic Diagram for the Simulation of an ECG signal

3.3.2 A Microcomputer System for Real-time Analysis of Cardiac Action Potentials

A system for performing real-time analysis of cardiac action potentials is developed using a microcomputer based on Motorola 6800 central processor by Sheridan et al [15]. The system can be characterized as follows

The microcomputer system used in [15]. to gather action potential data is based on the Motorola 6800 microprocessor. It contains 16k bytes of RAM and 8k bytes of erasable programmable ROM (EPROM) which holds the action

potential analysis(APA) programme. One channel of four-channel 8-bit ADC is used to digitize the action potentials. Digitized data is displayed on an oscilloscope via an 8-bit DAC. Program control is achieved using three switches connected via a second parallel port.

All software in [15]. are written in assembly language to obtain fast execution and, thus, high sample rates. They are stored in EPROMS to avoid the need for temporary storage facilities.

At the beginning, the signal is digitized at a rate of 12.5 kHz and the data is stored in a cyclic memory buffer which holds 12000 data bytes; the cyclic memory buffer is continuously updated. Simultaneously, the digitized data is reconverted to analogue form and displayed on continuously on the oscilloscope. The operator therefore sees the data being stored in real time.

When the operator wishes to analyze an observed action potential he/she switches the program to the 'display' mode in which the contents of a cyclic memory buffer are displayed on the oscilloscope. Data is displayed in blocks of about 6k bytes, representing a single sweep of the oscilloscope screen at 1cm/20ms. The action potential to be analyzed is defined by the operator using a potentiometer that supplies a voltage between 1 and -1 V.

When a satisfactory action potential is obtained the operator switches the program to 'analysis' mode, whereupon the action potential contained in the display block is analyzed for the action potential amplitude, the maximum rate of change of V_{max}, action potential duration at 50% and 100% repolarization and conduction time. On the completion of action potential analysis, the data and analysis are stored in memory for later printing. When 10 action potential analysis are performed, the results are automatically printed. figure 3.5 shows the three modes of operation which comprise the program.

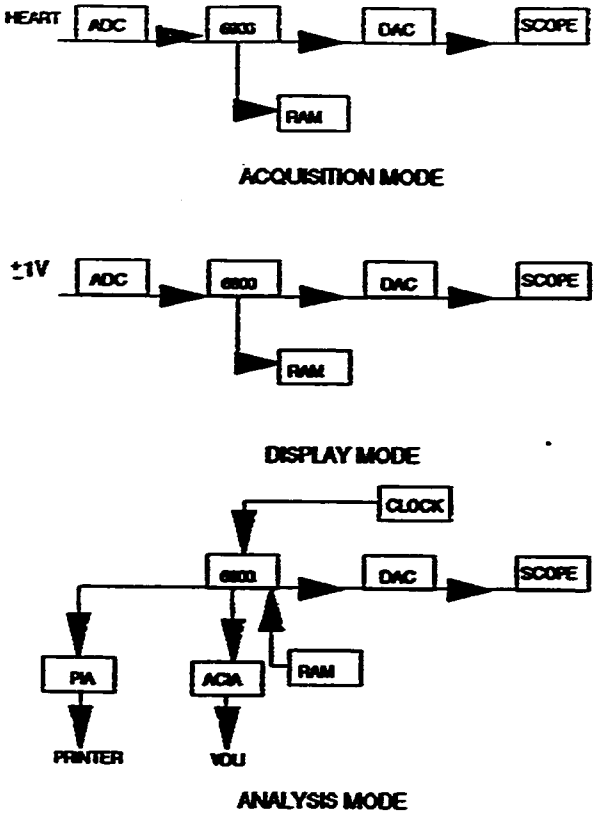


Figure 3.5: Software Modes for Action potential analysis

3.3.3 A Three-Channel Microcomputer System for Segmentation and Characterization of the Phonocardiogram

The phonocardiogram (PCG) is the instrument that detects the sounds and murmurs of the heart. Frequently, murmurs or alternations in the heart are the only definitive of some types of heart disease. Lehner and Rangayyan [16] developed a microcomputer system that segments the PCG and characterize murmurs using the ECG and carotid pulse as references.

The block diagram of the system in [16] that includes a 4 MHz MC68000 16-bit microprocessor, an Intel 8231 APU, 32K bytes of dynamic RAM and 16K bytes of firmware ROM is shown in Figure 3.6. Two MC6850 asynchronous communication interface adapters (ACIA's) provide the bus interface for two serial communication ports (RS-32 compatible). The system also has an MC68320 peripheral interface/time (PI/T) device.

A three channel data acquisition system including anti aliazing filters and a 12-bit A/D convertor is used to digitize the ECG, PCG and the carotid signals that are collected by the three channels. The low-pass filter cutoff frequencies for the PCG, ECG and the carotid signals are 500, 100 and 100 Hz, respectively. The signals are sampled at a rate of 1024 Hz over a duration of 1.5 s. Three DAC's are provided to display the various outputs.

The software in the system proposed in [16] depends on the digital signal processing techniques of the three signals. In the ECG signal useful information could be extracted from the detection of the onset of the QRS complex. The method used here is based on smoothed difference of the digitized ECG signal. The result is then used to find the RR interval. This in turn, in line with the output of the signal processing technique applied to the carotid signal, is used in analyzing the PCG signal.

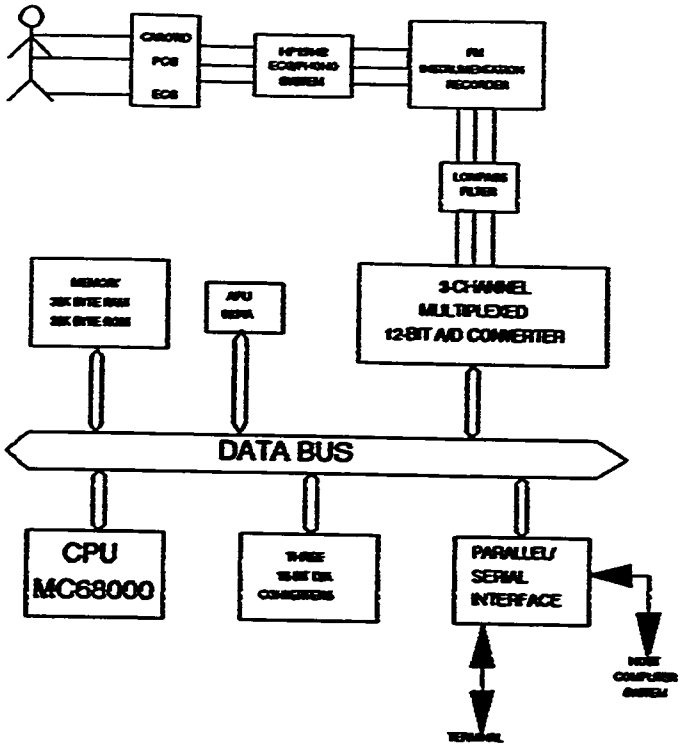


Figure 3.6: Hardware Block Diagram of the System

3.3.4 Other Microcomputer-Based ECG

Several other microcomputer-based ECG related structures have been suggested. Among those is a simulator of a 12-lead ECG developed by L. Shuqian and W. J. Tompkins [17]. It can be used as a standard signal for testing ECG (electrocardiogram) analysis systems. The simulated waveforms, which are based on Minnesota codes, are realistic looking ECGs. With this simulator it is possible to choose out of any one from the 100 Minnesota codes. Through an interactive interface, one can set the heart

rate (20 to 300 beat/min) and baseline drift (0 to 0.5 mV/s) and add random noise or 60 Hz interference. Since most of the samples are from typical clinical ECG findings, the simulated ECG not only meet the Minnesota code criteria, but also mimic clinical ECGs very well. It's possible to display all twelve leads on the screen, each six second long, or to direct the analog signals of any two selected leads through the output of the DAC interface card at sampling frequency from 100 to 500 samples/s.

Another approach is developed by M. Nasipuri, M. Kundu and D. K. Basu [18]. A low cost microprocessor based multiple holter tape processing system capable of monitoring four spools simultaneously at a speed up to 20 times real time is applied in this structure. Besides the normal holter scanner feature of continuously displaying the ECG waveforms at a faster rate than real time, it can accurately detect any possible abnormalities and highlight it.

A computerized bioelectrical cardiac monitor is developed by J. Jossinet et al [19]. It is concerned with the performance of impedance cardiograph. The impedance signals and the ECG are real time processed and four analog signals are displayed on the screen. A beat-to-beat esti-

mation of cardiac output and several indexes are computed and displayed. User's interface is by an interactive screen menus. Some facilities permit data data identification, storage and post-processing. The software has been designed in such a way as to be adaptable to each specific application and to enable the development of new routines for cardiac signal processing.

Another structure utilizes microcomputers in the care central stations, in which a four-bed central station that can be connected to any commercial intensive care unit (ICU) is developed in this research by F. D. Lucia et al [20]. The system is based on a personal computer as a local unit and on a microcontroller Intel 8031 as a remote unit. Four ECG signals are low-pass filtered, multiplexed, sampled at 256 samples/s, 8-bit A/D converted, preprocessed, and converted to serial format RS-232 by the remote unit. The real time display of the signals is at a standard speed of 25 and 50 mm/s. Heart rate, alarms, trend plots, and general patient data are shown on an Olivette M280 and EGA 13' color monitor as the local unit. Moreover, In order to reach standard monitoring sweep rates using a 13' screen with 640 multiplied by 350 pixels, an ECG data compression is implemented in the remote unit. This unit can support up to eight input

channels and can work with any PC via the RS-232, with the appropriate software. It can allow other software that may be developed, such as QRS detection, to be run under a PC command.

Microcomputers are also utilized in building expert ECG acquisition and analysis systems [21]. Developed by Perkusich et al, this research dealt with the design aspects of actual implementation phase of an expert system as an aid for the analysis and diagnosis of cardiac infirmities. An IBM-PC compatible microcomputer containing the necessary cards for the acquisition of ECG signals is used. The system provides electronic switching of the ECG lead configurations. A data and control flow diagram is used for the analysis of the software of the processed system. Particular attention is given to the signal acquisition, signal processing data, data management, and the man-machine interference.

3.4 Summary

In this chapter signal processing of the ECG signals is considered. All the DSP techniques described are used for the detection of the QRS complex onset because the QRS complex is the the most dominant wave in the ECG sig-

nal which makes it very important to get its features in the ECG signal processing. For interested reader, more techniques of signal processing can be found in the literature [8,32]. Moreover, some attempts of the use of microcomputers in cardiology are presented. It is worth repeating that all of the uses of microcomputers in cardiology haven't given attention to DSP capabilities available nowadays which we performed in research. This will facilitate to a great deal the expansion of DSP applications with ECG in the future. In the next chapter, we present the system proposed in thesis with detailed explanation. Moreover, the results obtained are shown, and the overall system performance is evaluated.

Chapter 4
THE PROPOSED SYSTEM: DESIGN, ANALYSIS AND PERFORMANCE EVALUATION

4.1 Overview

In the Chapter 2, the heart, its anatomy and function, the electrical behavior of cardiac cells, the ventricular cells and activation, the body-surface potentials, cardiac vector and bipolar and unipolar leads are presented. Moreover, in chapter 3, some of the applications of microcomputers and signal processing in cardiology are surveyed. In this chapter, the ECG with different stages of building an electrocardiogram is depicted. This leads us to the discussion of electrodes, biopotential amplifiers, Wilson resistors network, multiplexers, differential amplifiers and filtering. Moreover, an illustration of the system designed in this thesis and the results of laboratory tests are explained. This includes the description of both the hardware and the software of the

system. The hardware consists of the leads, interfacing unit, DSP board and the PC. The software explanation includes the description of a testing simulation on the main computer frame, the DSP32C software and the PC software.

4.2 System Implementation

At the start of our work, the DSP algorithms have to be tested before we start the implementation of our work. This first step is performed in order to know the credibility of the algorithms we are depending on.

4.2.1 Testing the Digital Signal Processing Software

In the early stages, it was very important to test the algorithms that are chosen for processing ECG signals. It's very important to notify that the main idea of signal processing in the methods chosen is the QRS detection. Testing is performed through simulation on a mainframe computer. Using an ECG signal from the output of an ECG machine. The simulation is made by sampling the signal manually. The algorithms are put in the frame of Fortran programs to obtain the results of testing. An extended method for detecting the onset of P wave and the end of T wave is tested, too.

All the methodologies discussed in Section 3.2 worked perfectly on the mainframe using Fortran language. However, the credibility of these methodologies was tested in the presence of white noise with a variance of 0.1 and 0.2. The signal is 18 V peak-to-peak. They proved to work perfectly under these circumstances. Figures 4.1, 4.2 and 4.3 shows the QRS onset detection, P wave onset detection and T wave end detection respectively. These figures show an indicator to the point that is detected in each of them. An indicator is shown on the onset of the QRS in Figure 4.1, on the P wave onset in Figure 4.2, and on the end of the T wave on Figure 4.3. All figures show the ECG signal corrupted with white noise. All methodologies that are described in Section 3.2 are tested for the detection of the QRS complex. Moreover, the algorithm that is based on amplitude and first derivative is tested for the detection of QRS, P and T waves. In addition to its simplicity, the extension of P and T wave detection in this algorithm urged us to use it in the real system. The Fortran programs are listed in appendix A.

In the next step, we build the interfacing circuit and then, the hardware and the software implementation are developed as follows in the rest of this chapter.

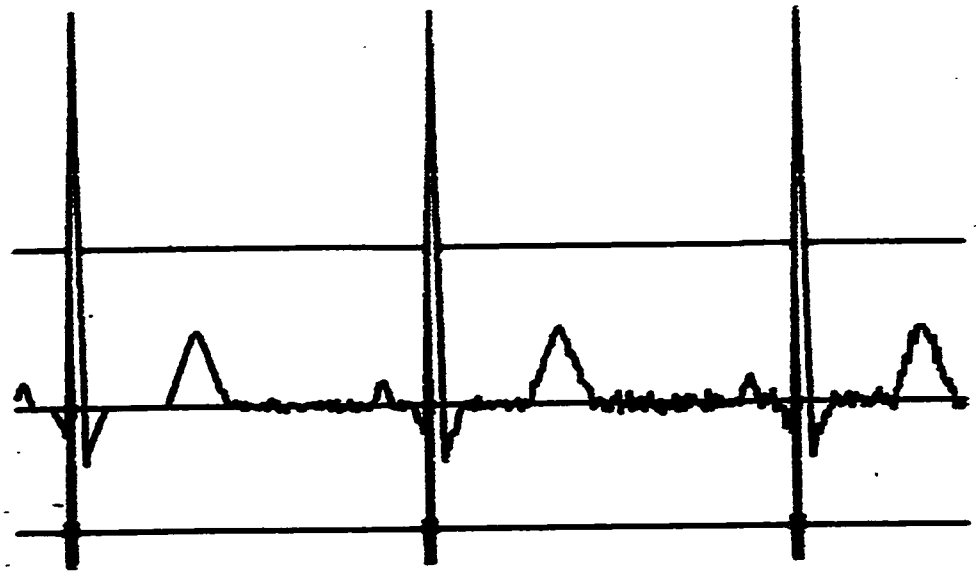


Figure 4.1: QRS wave detection

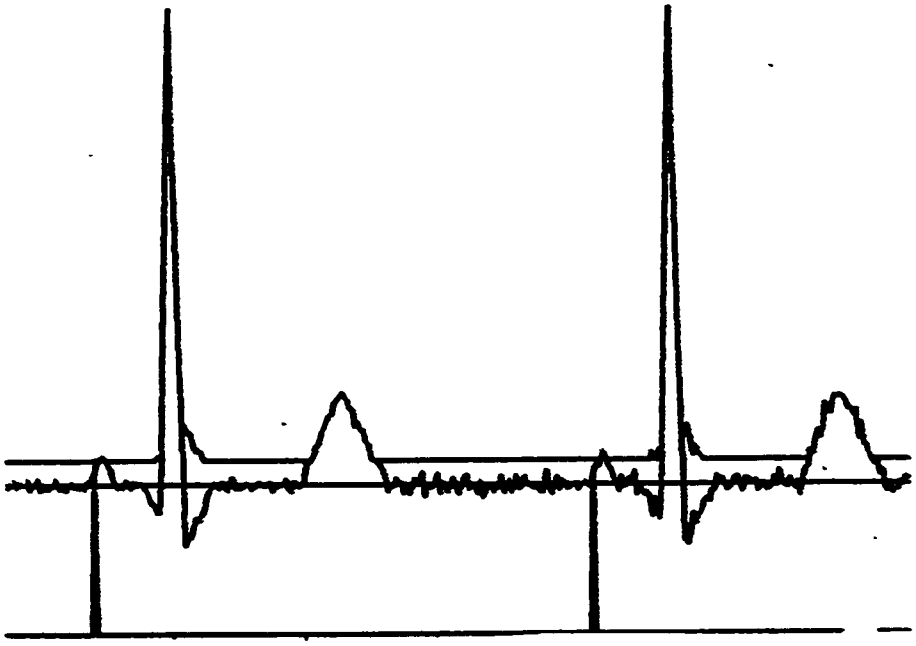


Figure 4.2: P wave detection

Figure 4.3: T wave detection

4.3 System Operation

The main objective of our work is to construct a DSP-based microcomputer system that can perform the same function as an ECG machine. In addition, real time processing can be provided by displaying various features which are extracted from the acquired system. Basically this requires performing several functions, namely;

1. multiplex several signals and amplify the selected signals;
2. condition analog signal from leads selected by the multiplexers;

3. digitize (sample) the signal to put it in a form of data;
4. process the data to extract useful information;
5. display the results.

The first and second functions is performed by means of input channels and amplifiers, third function is reached by analog to digital (A/D) converters, that act according to the principle of measuring data from an analog signal in specific time intervals. The use of the third function is to prepare the signal to be used by DSP board as well as the PC. The forth and fifth functions are performed by means of software analysis, on the DSP board and the PC, respectively. A schematic diagram of the overall system is shown in Figure 4.4.

Because of the above needs for our aim, the DSP-32C board is seen suitable for our application [22,23,24]. It's described in the following sections.

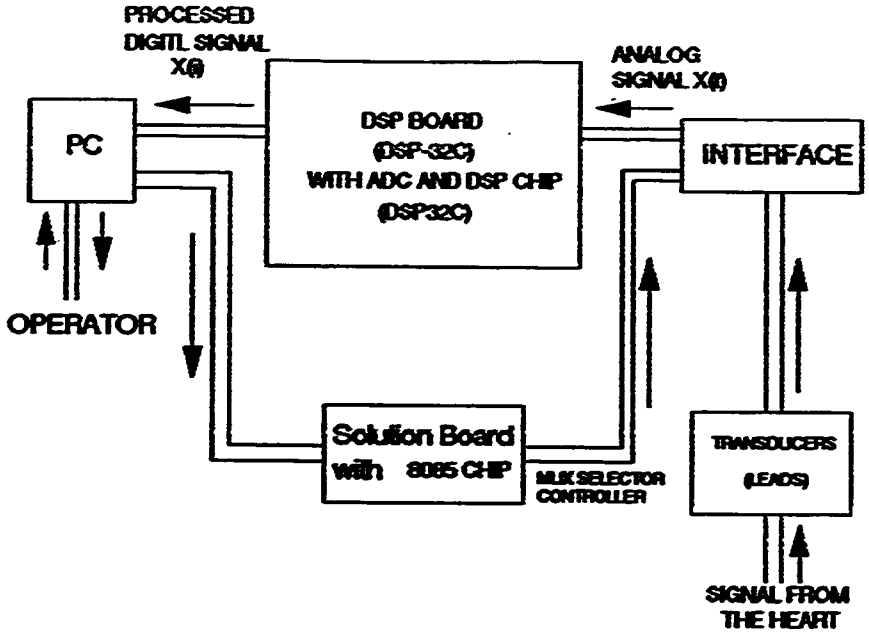


Figure 4.4: Functional Block Diagram of the Proposed ECG

4.4 System Hardware

According to the studies performed, the following system hardware is adopted as the logical design in order to achieve the results desired:

4.4.1 Electrodes

In order to examine and record potentials of the body, an interface should be introduced to the electronic device. This is why electrodes are used. This function for the first impression seems to be straightforward. Unfortunately, it is not. This is because currents in the

body are of ionic structure, i.e., currents are carried in the body by ions, while currents are carried by electrons in any electronic circuit. This means that a transducing function should be carried out by the electrodes. They must serve as transducers to change an ionic current structure into an electronic current structure [2,3,7].

4.4.2 The Electrode-electrolyte interface

The ***electrode-electrolyte*** interface is schematically illustrated in Figure 4.5 [2]. A net current crosses the interface passing from the electrode to the electrolyte consists of:

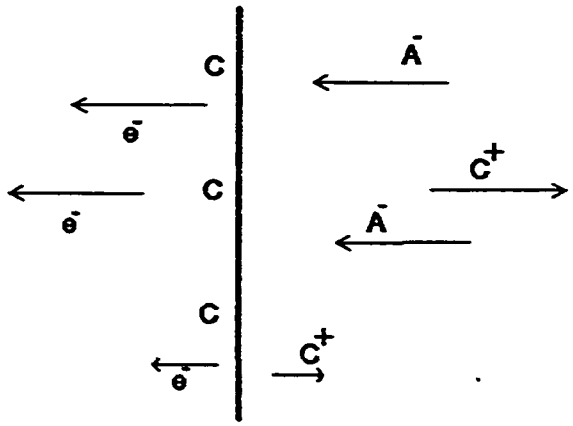


Figure 4.5: Electrode-electrolyte interface with current crossing from left to right

1. Electrons (e^-) moving in a direction opposite to that of the current in the electrode.
2. Cations (C^+) in the same direction as the current in the electrode.
3. Anions (A^-) moving in a direction opposite to that of the current in the electrode [2].

The electrode-electrolyte has a chemical reaction to give the transduction in order to transfer current from its ionic state into its electronic state. Moreover, the electrode possesses an electronic behavior that could be seen as shown in Figure 4.6 .

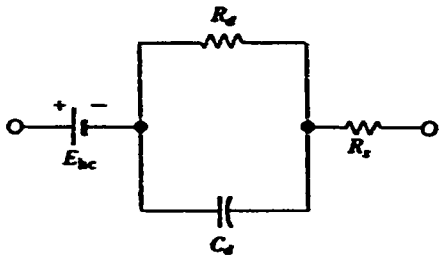


Figure 4.6: An equivalent circuit for an electrode

Many types of electrodes exist. We mention the polarizing, nonpolarizing and the silver-silver chloride electrodes here [2].

The transducer is the only component in the instrument process that contacts both the patient and the instrumentation. Six design parameters should be taken into consideration in order to make the best use of a transducer [2]. These design parameters are: sample loading, output impedance, damping, frequency responce and linearity. Each parameter is explained as follows.

Sample loading is the effect the transducer has on the system it is trying to measure [4]. Ideally, the transducer should not cause any change in the physiological or chemical parameter that drives the transducer. Because the driving source is mechanical or chemical, the transducer should offer the minimum mechanical resistance and cause minimal chemical activity.

The study of the ***output impedance*** of the transducer is important in terms of the match between the transducer and the next step (driven interface between the transducer and the processor). [4]

Because the voltage is the signal that we are interested in its transfer, the output impedance of the electrode should be smaller than the input impedance of the next stage.

Damping occurs when the transducer does not follow the physiological event faithfully. Three damping factors are present [4]

Underdamping:

Underdamping occurs when the transducers responds too quickly to a pulse-shaped physiological event resulting in an exaggerated, spiked, leading wave and a tendency to oscillate or ring on the trailing edge [4].

Overdamping:

Overdamping occurs when the transducers responds too slowly to a pulse-shaped physiological event resulting in a reduced, lagging wave [4].

Critical damping:

Critical damping is the desired situation where the response of the transducer will exactly follow the wave of the physiological event [4].

The damping effects is due to frequency response of the transducer. If the frequency response of it matches

the that of the physiological event, critical damping occurs. If its frequency response is greater than that of the physiological event, underdamping occurs. And if its frequency response is less than that of the physiological event, overdamping occurs. We can accomplish critical damping either mechanically or electronically. Electronic accomplishment of critical damping is done by placing a small variable capacitance with the electrode leads. It is adjusted until it resonates with the leads' inductance. The capacitance reactance then eliminates the inductor reactance because at resonance, the inductive reactance of the lead is equal but opposite in phase of the capacitive reactance of the lead. This will lead the electrode to appear as a perfect resistance which makes the electrode in the critical damping situation [4].

Frequency response of a transducer is the band of frequencies to which it will respond. As mentioned earlier, the frequency response an electrode is related to damping. If the electrode frequency response is less than the physiological event's bandwidth, then some information are lost. On the other hand, if it is greater some spurious information is sensed and a spurious results are added to the ECG.

Therefore, the frequency response of a transducer should be taken care of in designing an electrode in order for the electrode to be faithful in transferring the physiological event. Table 4.1 shows typical levels and frequency range of physiological signals [4].

Table 4.1: Frequency range of physiological events

	Typical Signal Range	Frequency Range
Heart potential (ECG)	50 μV to 5 mV	0.05 to 100 Hz
Brain Potential (EEG)	2 to 10 μV	1 to 100 Hz
Muscle Potential (EMG)	20 μV to 10 mV	10 Hz to 2 KHz (needle electrode) 10 Hz to 10 KHz (gross electrode)
Electro-oculogram (EOG)	10 μV to 4 mV	0.1 to 100 Hz

Linearity is desired in electrodes since it describes the situation in which the transducer output which is the physiological event electrical case, will track its input which is the physiological event ionic case.

The leads that are used in our hardware interface were Fukuda leads that have an input impedance of 10 M ohms and a max input circuit current of $1*10^{-8}$ Amperes. The leads have a CMRR of a 80 dB with a time constant of 3.2 sec. [25].

4.4.3 Signal Conditioning and Multiplexing Unit

The construction of this unit is made, taking into account that the ECG signal is very small, when it is collected by the leads. This forced us to include an amplifying stage at the front end to amplify the signals coming from the leads. Amplifying the signal is not performed through the first stage in the interfacing circuit only. Different stages of the interfacing circuit unit should accumulate for the desired potential amplification factor. A voltage gain of 200 (nearly 46 dB) or greater is desired. This is because the maximum potential the ECG signal with such an amplification factor will posses a value of 1 V or more. The 1 V value is satisfactory in our application for acquisition and display. In a study about the amplifying stage to the biopotential signals the following characteristics of amplifiers should be taken into consideration:

4.4.3.1 biopotential Amplifiers

The main characteristics of biopotential amplifiers are the following [2,26]

1. High input impedance
2. operate in a range of frequencies that meet the needs for frequency responses of biopotentials.
3. They should posses very high gain.
4. They should should have the capability of magnifying small amplitudes.
5. Deviation from linearity with less than 5%.

All of the above requirements would lead us to the use of operational amplifiers. Due to versatility and wide range of existence, operational amplifiers (Op-Amps) are used in our system. Their main task is to amplify the biopotential signal received from the leads. This first step needed the use of noninverting operational amplifiers (Op-Amps). The advantage of Op-Amps is that they offer a very large impedance at their input and very low impedance at their output. This allows a series of amplification stages with an overall gain calculated by multiplying the gains of different stages. Therefore, in designing our different stages an amplification factor of 200 should be taken into account. As can be calculated from Figure 4.7, the voltage gain of the non-inverting Op-Amp is [26]

$$A_{vf}=\frac{V_o}{V_i}=1+\frac{R_2}{R_1}. \qquad (3.1)$$

which makes our first amplification stage to be 7.667 if we choose $R_1=15k\Omega$ and $R_2=100k\Omega$. As a remark, all resistors used in the interfacing circuit are of 0.8 Watt rating. This makes them most suitable for the application sought relatively to the resistors found in the market.

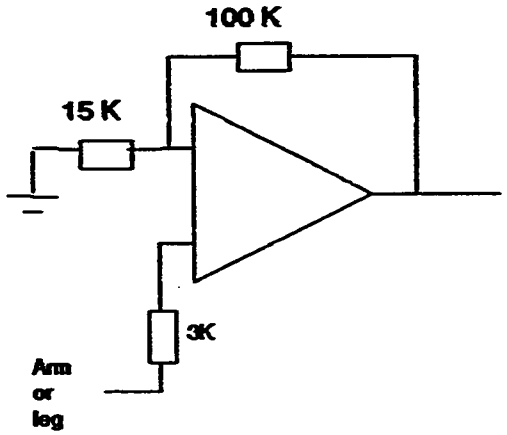


Figure 4.7: Non-inverting Op Amp used in the Interfacing Unit

A schematic diagram of the non-inverting Op-Amp used in our application is displayed in Figure 4.7. We can notice that a small impedance is placed in the input of the transducer so that a small loss in potential from the transducer is achieved.

Moreover, the patient should be protected from any flows of any leakage current. On the other hand, patient could be dangerous to the ECG monitor. For example, in a surgery, patients have their ECGs continuously monitored during the procedure. If the surgical procedure involves the use of another electrosurgical unit, which is usually the case, it can introduce onto the patient a relatively high voltage, which may in turn enter the electrocardiograph through the patient's electrodes.

These hazards could be overcome by driving all the output from the Op-Amps connected to the leads into an output of an Op-Amp connected to the RL lead which is in turn connected to ground. This will drive any leakage current into the output of the Op-Amp and minimize it to finally through it into ground. This scheme of protection is shown in Figure 4.8 and in Figure 4.9 the connections to the right leg (RL) is shown |2|

Another advantage of this protection method is reducing the effect of distortion introduced by the input ***common mode rejection ratio*** (CMRR). CMRR is defined as the ratio between the differential gain and the common-mode gain. It's not a problem in an ideal Op-Amp, because the difference in inputs in an ideal Op-Amp is 0, which makes

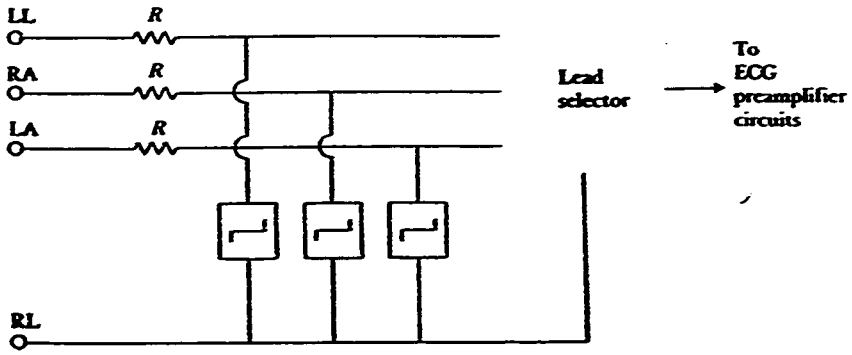


Figure 4.8: **Protection scheme of the leakage current for CMRR**

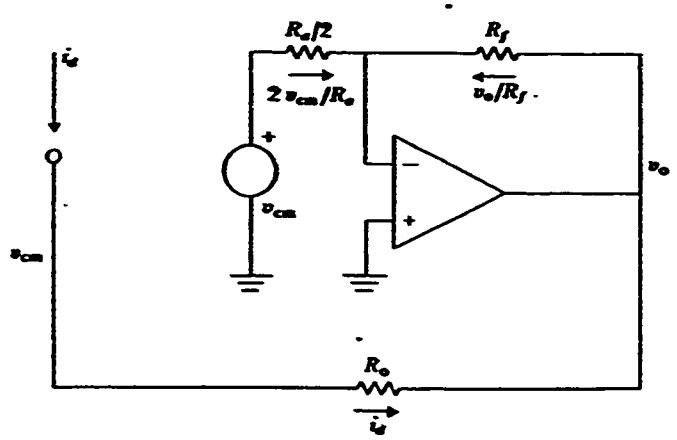


Figure 4.9: **Electrical circuit used for protection of the leakage current**

the differential gain equals to 0 and hence CMRR equals 0

[2]. However, in practical Op-Amp the the differential gain input is not exactly 0, and CMRR occurs. It might be small, but it causes an additional noise to the original signal. This is overcome by driving the leakage current that causes this noise into the RL lead. This minimizes the CMRR.

4.4.3.2 Resistors Network

Readings of leads I, II, III are given by the equations given before in chapter 2

1. I = LA - RA
2. II = LL - RA
3. III = LL - LA

where RA, LA, and LL are the readings of the potentials read from the right arm, left arm and left foot electrodes respectively. [2]

The leads vectors for these leads is shown in Figure 4.10

Readings of leads aVR, aVL and aVF are simulated in the electrical circuits shown in the Figures 4.11, 4.12 and 4.13 .

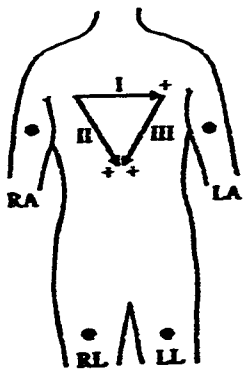


Figure 4.10: Lead vectors for standard limb leads

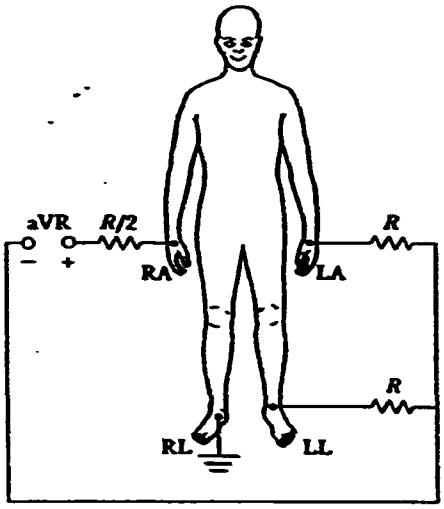


Figure 4.11: An equivalent electrical circuit of the lead aVR

The Wilson network had to be installed on the interfacing circuitry. The use of Wilson network is to give

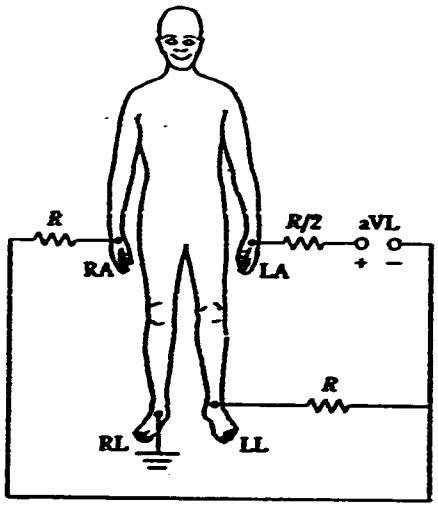


Figure 4.12: An equivalent electrical circuit of the lead aVL

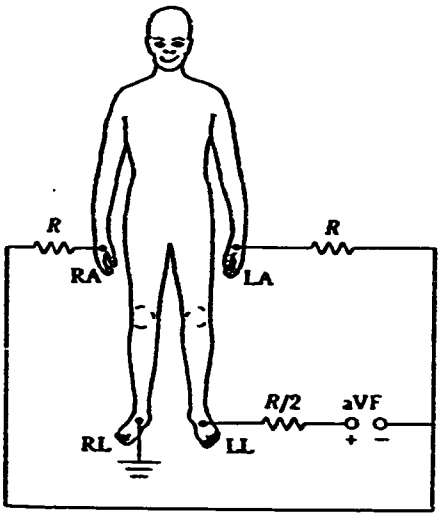


Figure 4.13: An equivalent electrical circuit of the lead aVF

the necessary resistance in order to generate the limb ECG readings. The Wilson network consists of two different values of resistors, R and R/2. This will give us the ability of producing all of the limb leads. In Figure

4.20 a display of Wilson network as a part of the whole circuit is shown.

In the following stage, two multiplexers are placed in order to be able to select the readings from the leads. This is necessary because each of the ECG signals is the difference between a pair of the electrodes readings. Two RS DG-508 CMOS multiplexers are used in our application. They are 8X1 multiplexers. The use of this unit is because of the need of a multiplexer with an input that contains more than 6 bits as an input to it for the use in multiplexing the arm leads (see Figure 4.14).

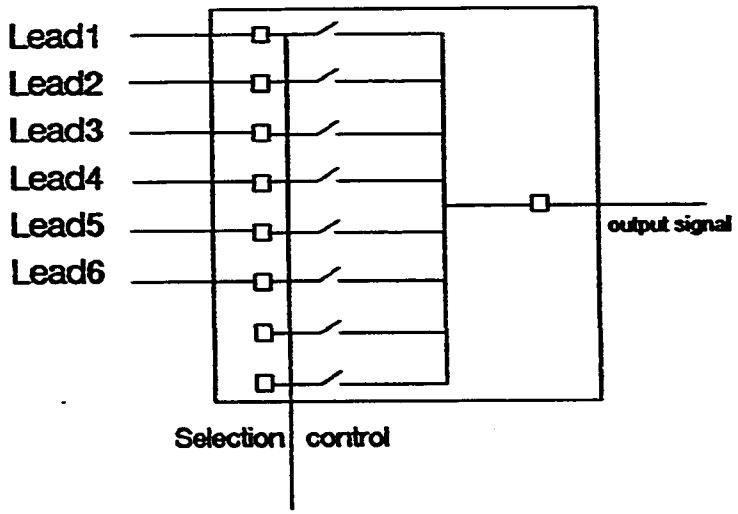


Figure 4.14: A schematic Diagram of a Multiplexer

The next step of the interfacing circuit is the differential amplifier. The output of the multiplexers will be fed to the input of the differential amplifier stage (see Figure 4.15). This will result in an output from this stage that meets the following gain equation:

$$A_{vf}=(1+\frac{2R'}{R})\frac{R_2}{R_1}(V_1-V_2). \qquad (3.2)$$

This will give us a gain of 3 if all resistors are equal. This will introduce an amplifying factor of 23 at the end of this stage.

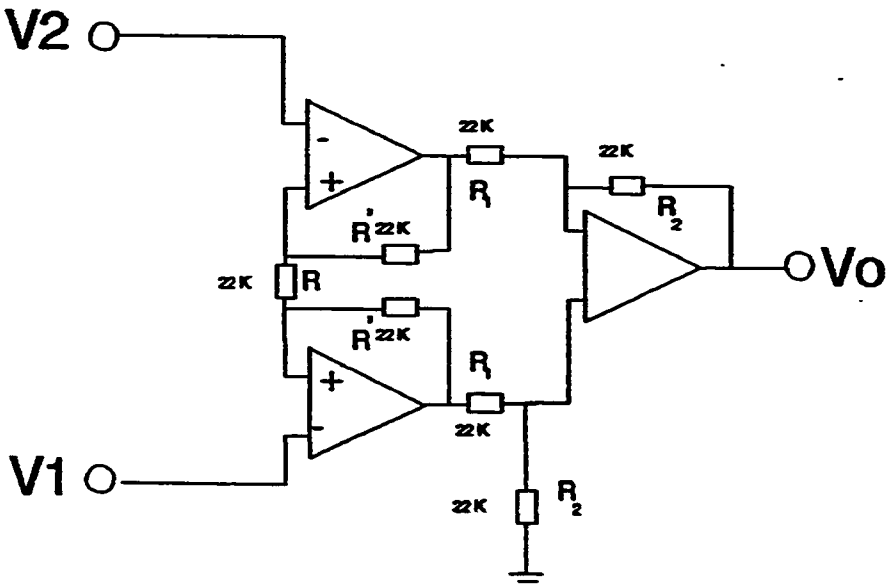


Figure 4.15: Differential Op Amp used in the Interfacing Unit

As noticed, the output of this stage will be the amplified signal of the difference between the two inputs to it. The reason behind not using only one Op Amp in this stage is that we need a very high input impedance to satisfy the fact that we have very low potential. This scheme of differential amplifier supplies very high input impedance in this stage of the circuit.

Power line noise occurs because of the frequency encountered in power (60 Hz in Saudi Arabia). It affects the shape of an ECG signal. This is because of the comparability of the frequencies of the ECG signal and the power line signal. This frequency interferes with the ECG recording and corrupts it. The next stage installed in the interface circuit is a 60 Hz notch filter. This filter rejects the 60 Hz power line noise. It is enough to build a second order notch filter to achieve good rejection of the 60-Hz power line noise and not affect the shape of the ECG signal. Figure 4.16 displays the Notch filter with the design parameters shown. Figure 4.17 shows the frequency response of the Notch filter built in the circuit [26].

In the design of the Notch filter, a practical way is to choose $R_1 = R_3$ equal to a standard value greater than 100 times the output resistance of the source. Then we set

$R_2 = \frac{R_1}{2}$ and

$$C_1 = C_3 = \frac{1}{4\Pi f_0 R_2} \tag{3.3}$$

where f_0 is the cut-off frequency. Then set $C_2 = 2C_1$[24]

Choosing

$$R_1 = R_3 = 2M\Omega$$

then

$$R_2 = \frac{R_1}{2} = 1M\Omega$$

$$C_1 = C_3 = \frac{1}{4\Pi(60)10^6} = 1320pF$$

$$C_2 = 2C_1 = 2640pF$$

The last step before feeding the signal into the built-in ADC of the digital signal processing board (DSP32-C), is the building of a 150 Hz low pass filter (LPF) that will reject the noise that is coming from any electronic devices that are around the machine. The LPF will reject any frequencies that are higher than 150 Hz.

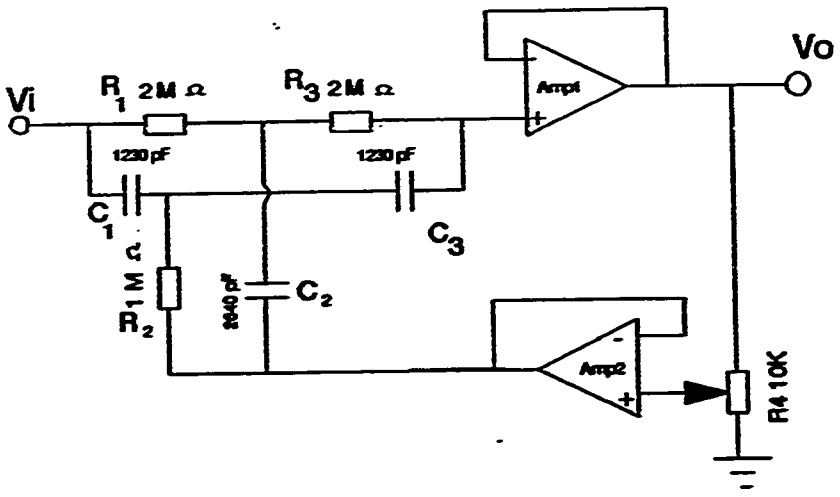


Figure 4.16: Notch Filter used in the Interfacing Unit

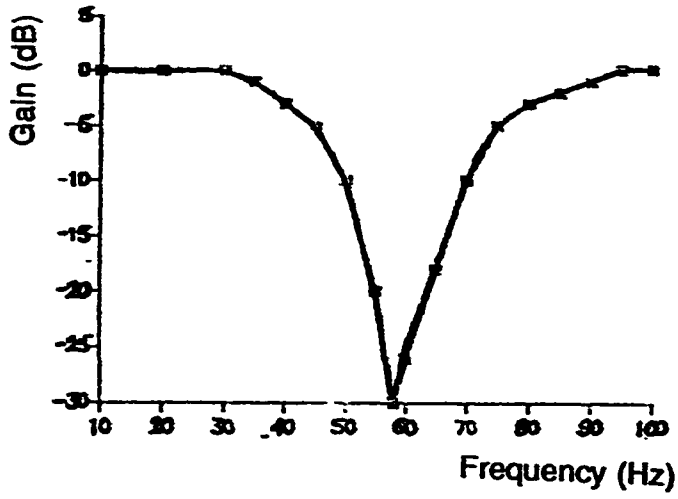


Figure 4.17: Frequency Response of Notch Filter used in The Interfacing Unit

In the design of the of the low pass filter a simple one is chosen. This is the voltage-controlled-voltage-source or VCVS filter. Figure 4.18 shows the schematic diagram of the LPF in our circuit [26]. The cut-off frequency of this filter is given by

$$f_c = \frac{1}{2\Pi(R_1 R_2 C_1 C_2)^{1/2}}$$

Choosing $c_1 = 2c_2 = 0.04\mu F$ and $R_1 = R_2 = 14.9K\Omega$ then a gain of about 8 and a cut-off frequency of about 100 Hz will occur. Figure 4.19 shows the frequency response of the LPF.

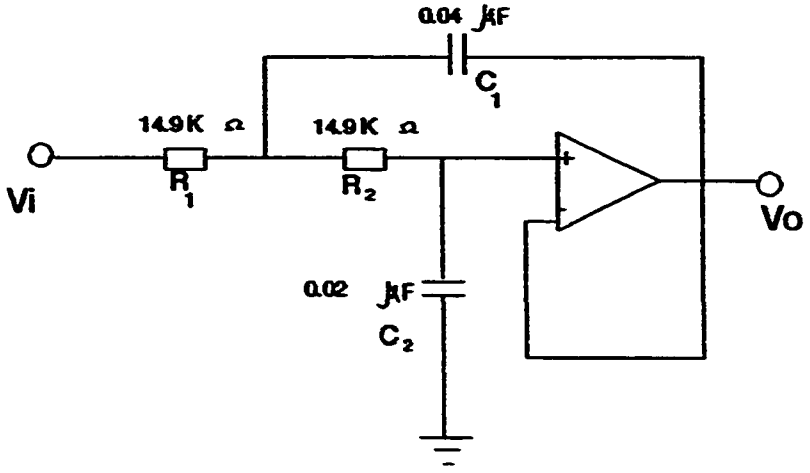


Figure 4.18: Low Pass filter used in the Interfacing Unit

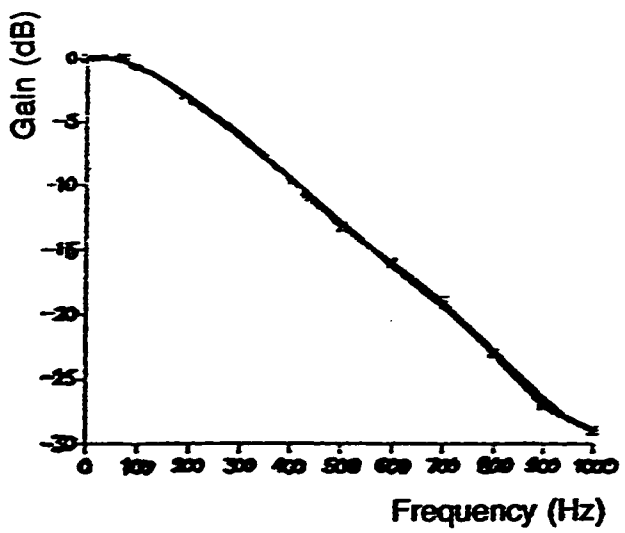


Figure 4.19: Frequency Response of LPF used in The Interfacing Unit

By this, the interface circuit is built and what remains is the work on the DSP32C and the DSP32-C. In Figure 4.20 The complete circuit diagram of the interfacing unit is displayed.

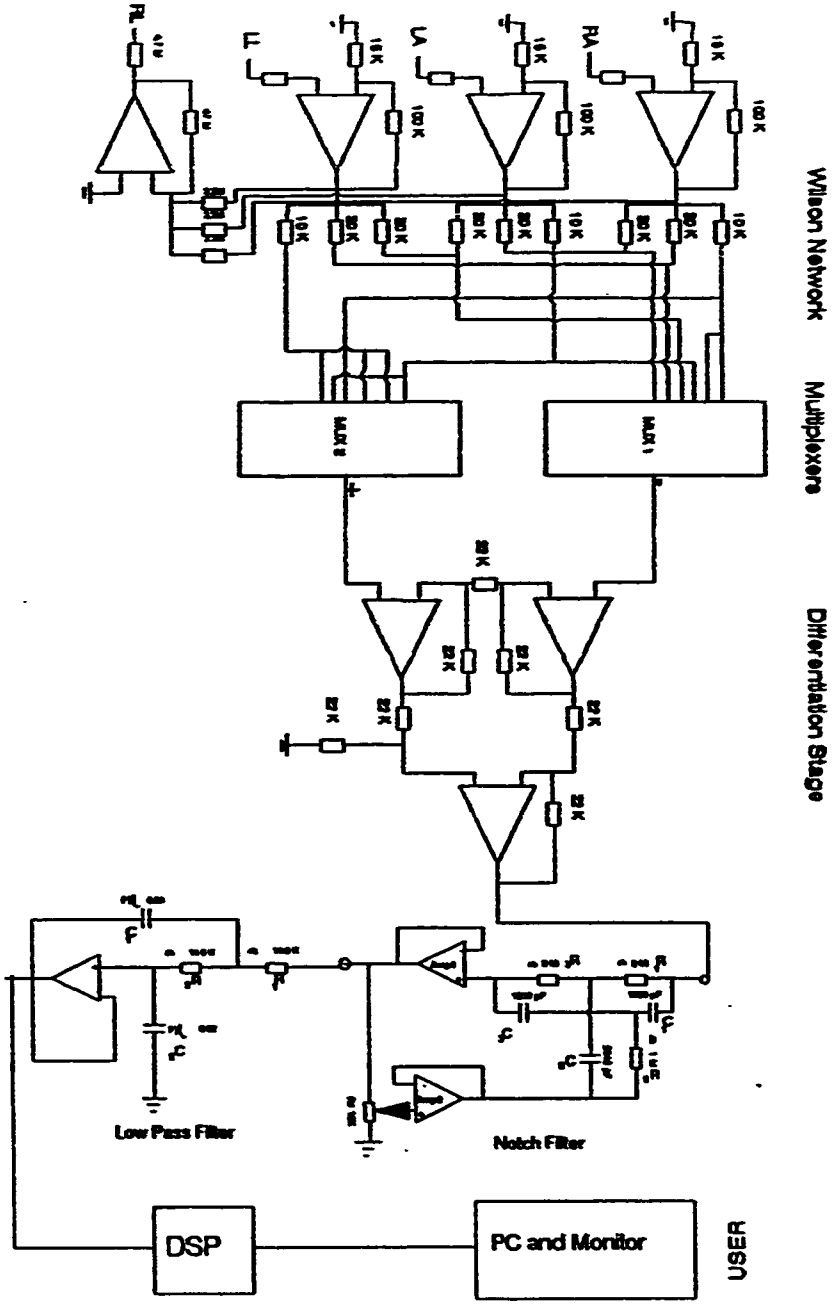


Figure 4.20: Circuit Diagram of the Interfacing Unit

4.4.4 The DSP board and The DSP32C chip

Many alternatives are given in our choice of the digital signal processing board to be used in our system. The DSP32C chip was our preference to be used in the system due to the facilities it offers with the DSP Ariel board it is installed on. The chip is a 50 MHz AT&T DSP chip with many other features that coincide with the reputation of AT&T company [23]. Moreover, Ariel Company supplied the DSP-32C board that the DSP32C chip is installed on by a "C-language" compiler that simplifies the programming of the chip and board. However, it is very important to notice that our choice is three years old which makes it unfair to compare it with the Technology available nowadays.

4.4.4.1 General information about the DSP-32C board

The DSP-32C board is a single-board DSP system for IBM-compatible personal computers. It is manufactured by Ariel and uses the AT&T DSP32C [24] chip as its main processor. DSP-32C notation is to be used throughout the rest of this chapter and the following chapters to refer to the DSP Ariel board, and DSP32C notation to refer to AT&T DSP chip.

The DSP-32C is composed of a DSP32C chip and DSP-related resources, which are under the control of the DSP32C. A brief description of the DSP-32C components is given as follows [23]:

1-DSP32C Processor specifications

1. 50 MHz AT&T DSP32C
2. 80 ns instruction cycle time
3. 25 MFLOP peak performance
4. 32-bit floating-point arithmetic
5. Single-precision IEEE floating-point compatibility
6. 16- and 24-bit integer operation
7. 16 bit serial I/O port and 16 bit parallel I/O port with DMA options.
8. 1536 words of on chip memory
9. 8- or 16-bit microprocessor host interface
10. internal and external interrupts
11. Low power (.75 μm) CMOS technology.

The major subsystems of the DSP32C include a control arithmetic unit (CAU), a data arithmetic unit (DAU), on-chip memory (RAM only or a RAM/ROM combination), a parallel I/O port (PIO) and a serial I/O port (SIO).

The execution unit of the device are the CAU and the DAU. They are used to achieve the high throughput of the device. The CAU which is for the performance of branching, data transfer, fixed-point arithmetic, logic operation in parallel with the DAU operations, and generating addresses for the DAU operands, performs 16- or 24-bit fixed point arithmetic for logic and control functions. Integrating with the CAU, the DAU performs 32-bit floating-point and data type conversion operations [23].

Except for a register load, execution of a CAU instruction is completed before the next instruction begins execution. This simplifies using the CAU for logic and control operations. The DAU, on the other hand, employs a four-stage pipeline to perform 25 million floating-point computations per second. This means there are a small number of latencies associated with the DAU's instruction set [23].

The DAU is configured for multiply-accumulate operations and is the primary execution unit for signal processing algorithms. It contains a floating-point multiplier and adder that work in parallel to perform computations of the form $a = b + c * d$. To understand the operation of the DSP32C, one must understand the pipeline operation of the DAU. The DAU employs a straightforward

fetch-multiply-accumulate-write pipeline. Briefly, the DAU executes a multiply-accumulate instruction in four stages: fetch of c and d, multiplication of c and d, accumulation of the c and d product with b(with the result stored in the accumulator), and an optional write of the result to memory or an I/O port. A maximum of two of the three multiply-accumulate operands come from memory [23].

Data is transported throughout the device via a 32-bit bus. The data bus supports memory accesses during each instruction cycle: an instruction fetch, two operand reads, and a write to memory. This high-speed data bus and the pipelined architecture of the device allow the DSP32C to fetch two 32-bit operands from memory or an I/O port, perform multiply and accumulate operations, and write a result to an I/O port or memory during each instruction cycle [23].

The DSP32C provides on-chip memory and an external memory interface for off-chip memory expansions and memory mapped peripherals. Two on-chip memory options are available. The first supplies three 512-word RAM's on-chip. The second option replaces one 512 word RAM with 2 Kwords of ROM. A 24-bit addresses capability increases

the external memory capability to 16 Mbytes. Programs can treat memory as a common resource, with instructions and data arbitrarily residing in on-chip RAM, on-chip ROM, or external memory. The external memory interface supports wait-states and bus arbitration.

DSP32C Memory Mapping

The DSP32C has 8 memory modes that are selected by three control pins on the device. These modes selects the arrangement of memory resources within a 16 Mbyte address space (24-bit addresses).

The make utility found in the device controls the memory mapping of the DSP32C executable (output) files according to the physical memory mapping of the DSP32C.

Memory modes 0-3 of the DSP32C provide compatibility with the DSP32. However, the memory modes 4-7 utilize the full 16 Mbytes address space provided by the DSP32C device internal memory. The memory mapping of the DSP32C contains different resources, namely, internal and external memory.

When the programmer wishes to specify the memory mapping of the data and the text of the software he/she is writing, he/she can use the directive **.rsect** in order to specify the section of memory that he/she would like to

load his/her data or text in. The .rsect directive could control the loading of the data or text in the following sections:

1. ram0: which includes the on-chip RAM 0 with a maximum size of 2 Kbytes
2. ram1: which includes the on-chip RAM 1 with a maximum size of 2 Kbytes
3. ram2: which includes the on-chip RAM 2 with a maximum size of 2 Kbytes
4. extA: which includes the external memory A with a maximum size of 6144 Kbytes
5. extB: which includes the external memory B with a maximum size of 10232 Kbytes

For more details on the physical location of the different sections of DSP32C memory, please refer to appendix F that includes all the memory maps for each memory modes for the DSP32C. For further details, see reference [27], and the PC interface section later in this chapter.

The parallel I/O (PIO) port provides a parallel interface for communication for the DSP32C and external devices. It may be configured as an 8-bit (DSP32 compatible) or 16-bit port. The serial I/O (SIO) port provides serial communication and synchronization with devices

outside the DSP32C. Three on-chip direct memory access (DMA) controllers support direct memory access via the serial input, serial output and parallel I/O ports. A single level interrupt facility can respond to four internal and two external, individually maskable sources. A relocatable vector table controls program flow based on the source of the interrupt [24].

The DSP-32C board makes the integration capabilities that enables the DSP32C chip to interact with the surrounding environment. The DSP-32C board augments the functions of the DSP32C chip by providing these additional capabilities [24]:

1. 64 Kbytes zero-wait state RAM standard, 256 Kbytes optional.
2. Two channel of 16-bit analog I/O, including high performance I/O stages and antialiazing filters.
3. An industry standard SCSI disc drive interface.
4. DSPnet, a versatile, multimaster 32-bit wide expansion bus for interconnecting multiple DSP boards.
5. single bit auxiliary I/O through rear panel.

The rest of this section will summarize the features of the DSP-32C board peripherals.

4.4.4.2 Analog Input

Two channels of 16-bit analog to digital convertors, using high quality, professional audio connectors. The input have differential input stages for CMRR with overvoltage/surge protection circuitry. The input channels are sampled simultaneously (within less than 10 ns). Crystal controlled sample rates are software selectable from 2 KHz to 100 KHz. A single 12-bit, 400 KHz sampling rate channel is also available. In ECG application, however, we use sampling rate of 2 KHz. In order to process the signal properly, a duration of 4 seconds is required. This requires a huge space of memory. Since the bandwidth of ECG signals is limited(200 Hz), we perform subsampling decimation, where we skip 7 samples for each sample taken. This makes the effective sampling rate to be 250 Hz.

4.4.4.3 Analog Output

Two channels of 16-bit digital to analog convertors with fixed 20 KHz 9th order elliptic reconstruction filters and sin(x)/x compensation is available. In our application the analog output is not used. Note that the analog output is not used in our system, however, it might be of use in any future research.

4.4.4.4 PC Interface

The DSP-32C may be installed in any 8- or 16-bit PC compatible slot. It consumes 32 contiguous 8-bit or 16 16-bit I/O ports mapped by a six-position DIP switch. The addressing of the DSP-32C is performed by using the address switch [26]. In this project we set the beginning address of the board at 0x320 hexadecimal (0x320H). This implies that the board occupies the next 32 memory addreses, i.e., from 0x320H to 0x33EH.

In addition to the DSP32-C board, another board is installed on the PC. This is an I/O interface board that uses the 8255 chip and called scientific solution board. This board is used in order to set up the multiplexers for a specific value in order to give us the different readings of the ECG, under PC control.

The next step after building the interface circuit is to set up our PC and the board installed on it. The job here is the development of software. It is now about the study of how to supply the software necessary for data acquisition, digital signal processing, graphical representation and interface between the PC and the DSP32-C board.

4.5 System Software

In this section a description of all sets of software developed in this thesis will be illustrated. General information about our needs of the software will be given first. Later, explanation of the DSP32C software algorithms is described. Finally, the PC software is depicted.

4.5.1 General Information

There are usually two programs to design an application for the DSP-32C : high-level-language program for the PC and a signal processing program for the DSP32C.

The high-level-language PC program is utilized to execute two main functions. First, make an interface between the user and the board. Second, download the signal processing program to the board. However, the high-level PC program could be omitted if there is no information to be given by the DSP-32C signal processing program to the user, or no instructions are to be given in the reverse direction.

The PC program could be written by any high-level language. However, C language is the manufacturer and our choice. One advantage of C language is that input and

output functions calls may be declared "intrinsic". This tells the compiler to embed the input or output instructions in line rather than calling them as functions. This provides assembly language-like performance for the low-level devices [28,29,30].

The signal processing program executes the signal processing algorithm (described in section 3.2) and handles analog to digital (A/D) and digital to analog (D/A) functions.

The signal processing program can be written using DSP32C assembly language. However, when C language with the AT&T compiler is used, some interfacing with the assembly language is required when Input output routines are built.

By using the software many hardware resources can be defined. The following are parts that are used in this thesis:

4.5.1.1 Mode Selection

The DSP-32C analog interface section supports two modes of operation. Normal mode at which the board provides one or two channels of 16-bit ADC with a sample rate between 2 KHz and 100 KHz, and DAC with a sample rate of 20 KHz. The high speed mode gives the ability of making ADC with sample rates that range from 8 KHz to 400 KHz. In the high speed mode we sacrifice in the terms of that we lose one of the two channels and the operating channel is a 12-bit channel. Moreover, the DAC is not in operation in the high speed mode. As mentioned earlier, this work needs the selection of the 2 KHz sampling rate.

4.5.1.2 I/O control

An introduction to serial I/O (SIO) is given here because it is used in the thesis. Three types of SIO are configured:

1. Program-controlled SIO (polled)
2. DMA-controlled SIO
3. Interrupt-controlled SIO

The main software resources for SIO programming is some flags that state if the status of the input buffer, output buffer and synchronization. In this

thesis, we chose the DMA-controlled SIO to be used in acquiring data from the leads to be displayed. However, the polled SIO technique is selected for the digital signal processing of the ECG.

4.5.1.3 Language Interface

As mentioned earlier, two programs could be used acquiring the services of the DSP-32C. This requires that the high-level PC language needs some interfacing with the DSP32C signal processing program. This occurs by accessing variables with information such as DSP type, I/O address from the signal processing program to the PC program. Therefore, at the beginning of the C high-level PC program a header file that declares some variables should be included by the statement [23]:

```
#include <dsputil.h>
```

This file is found in the include libraries of the software accessories that are given with the board.

Moreover, routines that are used to upload/download the information from/to the PC program to/from the DSP32C signal processing program are found in libraries called (dsputilx.lib), where x denotes the memory

model that is used in the program. It can be l (large memory model), m (medium memory model), or s (small memory model) which is the default one.

4.5.2 Digital Signal Analysis

In this section the software that is built for the ECG signal analysis and feature extraction is introduced. This is performed in different step that are described now.

4.5.2.1 DSP Software

The next step is to translate these programs into C language. Again, these C programs had to be tested before they are used in a real system. By using the built-in simulator in the DSP32-C, they were examined and they gave very good results checking for the onsets of the QRS and P waves, and the end of the T wave.

The DSP32-C board can be programmed in C language, and the interfacing terminology was also supplied by C language. The DSP32-C board was programmed for two purposes: data acquisition and digital signal processing. The data acquisition part is done using polled serial I/O method. This method, although might be time consuming rather than the DMA (direct memory access), it is used because of its simplicity [24].

The main idea of polled input is checking the status of the input buffer flag. If it indicates that the input buffer is full the input is polled and sent to memory. Otherwise, a delay waiting loop will wait till the buffer is full. The main source of programming this section is assembly language Macros because the C language doesn't have the power of checking the status of the registers in the DSP32-C board.

The programming scheme of this stage would look as follows:

In the C-language program a buffer is assigned to a function that is defined in the assembly program. A counter is set to 0 and then the buffer is filled in data unti the counter reaches the number of samples desired to be contained in the buffer.

in the assembly program a function is defined as a global one and the input buffer of the DSP32-C is checked if it is empty or full. If it is empty, a delay loop is activated till it is full. When the input buffer is full, the value that it contains is assigned to the global function which is in turn assigns the same value to the buffer in the C-language subroutine (see Figure 4.21).

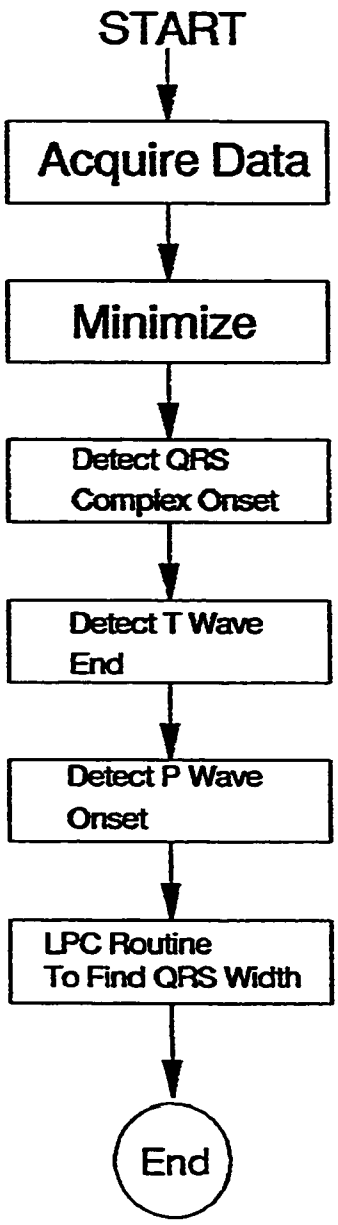


Figure 4.21: DSP32C Software Block Diagram

When the main program calls the subfunction get_data, the value that is found in r1 register is fed directly into the variable that is equated to the subfunction, i.e.,

```
variable = get_data();
```

The next scheme of programming the DSP32-C board is the digital signal processing part. In this part, the different methodologies described in the literature review are translated into C language software. In order to make sure that the methodologies work in a satisfying way, an ECG signal is simulated from a given ECG output, then these methodologies are applied in order to detect the onset of the QRS of an ECG signal. Moreover, a newly created method based on the amplitude and 1st derivative algorithm is tested. This methodology is beneficial for detecting P wave onset and the end of the T wave. These measurements are needed because the following intervals should be measured:

1. RR:The measurement between two R waves
2. PQ:The measurement between the start of P wave and the start of QRS wave
3. QT:The measurement between the start of QRS wave and the end of T wave

Moreover, the method of using linear prediction for feature extraction of the QRS wave, was used to extract the width of the QRS wave. Results are shown later in this chapter.

4.5.2.2 DSP-32C Software Organization

The DSP-32C board can be programmed in two languages namely, DSP-32C assembly language and "C" language. Two programs are written in our application. Because of efficiency, the data acquisition is performed by means of assembly software. This software consists of several modules (refer to Figure 4.21). They can be described as follows,

1. Initialization module sets the value of Processor Control Word (PCW), Input / Output Control Register (IOC) and sampling rate (Figure 4.22) Moreover, the arrays of data are designed in this module as global.
2. The data acquisition module specifies the direct memory access (DMA) method as the procedure to be used for data acquisition. This gives a very efficient and fast way to acquire data in real time.

3. When all the data are acquired, a flag is set telling the PC that it can upload the data from the DSP-32C memory to the PC memory.

Moreover, because of the simplicity and efficiency too, "C" language is used in the digital signal processing software. It consists of the following modules;

1. Set the sampling rate, the IOC and the PCW of the device. This is done by using assembly macros(Figure 4.22).
2. Acquire the data from lead "II" in a range of four seconds. This is again performed by means of assembly macros(Figure 4.23).
3. Because of DSP32C RAM constraints, a minimizing module is built so that it makes the sampling rate one eighth of the original one(Figure 4.24).
4. A QRS onset computation is then carried out in the following sequence(Figure 4.25);
    a. The absolute value of the ECG data is calculated for each sample.
    b. The maximum of the ECG signal is calculated in another module.
    c. A threshold is set with a value of 0.4 of the maximum of the signal.

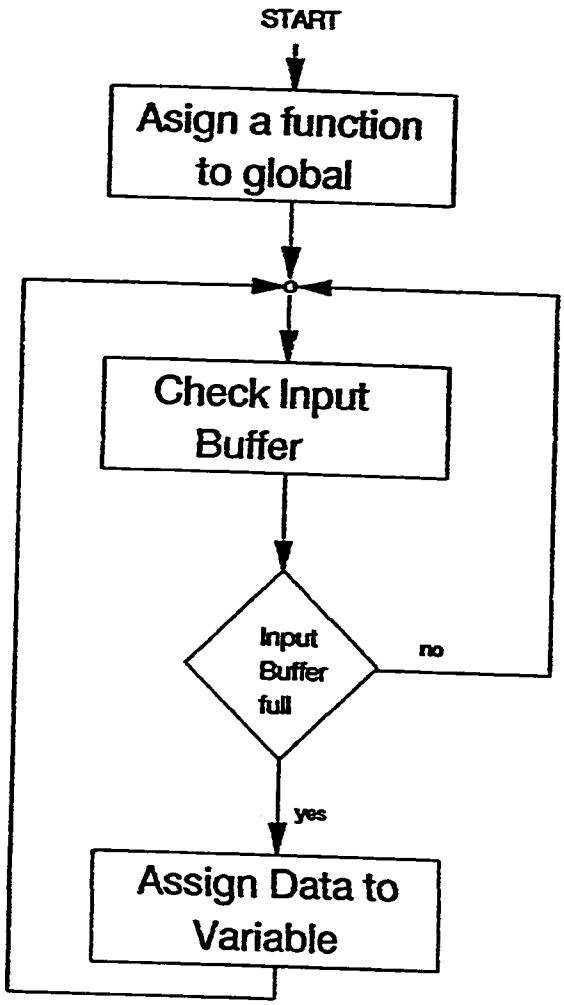


Figure 4.22: Assembly Routine For Data Acquisition

d. A clipper is built so that the absolute value of the signal equates itself if it is greater than the threshold and equates the threshold value otherwise.

e. The output of the clipper is differentiated by means of a first derivative.

f. The derivative is then compared with a constant threshold. A candidate for a QRS onset occurs when the derivative is greater than this constant threshold.

5. The computation of the "T" wave end is then performed as follows (Figure 4.25);

a. Calculate the threshold as one tenth the maximum of the signal.

b. Pass the absolute value through a clipper.

c. Starting after the QRS signal, the end of the "T" wave is calculated by specifying its onset by comparing the derivative of the clipped signal with a threshold of 0 value. After that, the first 0 valued derivative is considered to be a candidate of "T" wave end.

6. The "P" wave onset is computed according to the following (Figure 4.25);

a. calculate the threshold as one twentieth the maximum of the signal.

b. Pass the absolute value through a clipper.

c. Starting after the "T" wave, the onset of the "P" wave is calculated by specifying its onset by comparing the derivative of the clipped signal with a threshold of 0 value.

In programing the DSP32C board, two types of data acquisition techniques are adopted. One is considered with getting data from the leads in order to display it. In this, the integer type of data is used. This is because it makes no difference in overflow. However, because many calculations have to be done in the signal processing programs, the data had to be acquired in the floating point arithmetic. In turn, the Input/Output control (IOC) register has to be set to a specific word. This assignment will take into consideration the following:

1. Using float (y) instruction for converting the input buffer from an integer to a float number will force us to use a 32- or 16-bits of input. Moreover, the input should reside in the upper 16 bit of input buffer (ibuf).

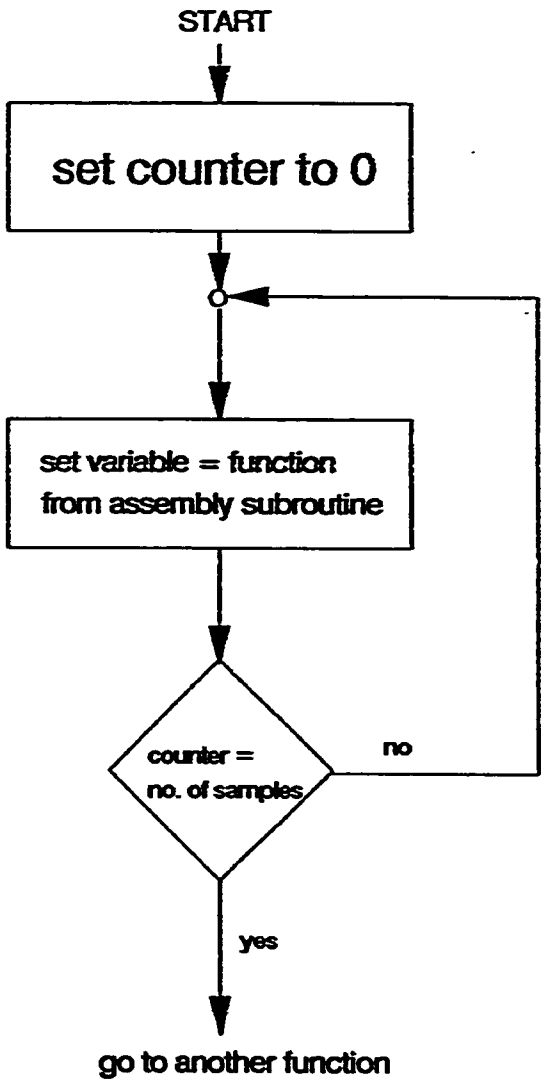


Figure 4.23: C-language Function (get_data)

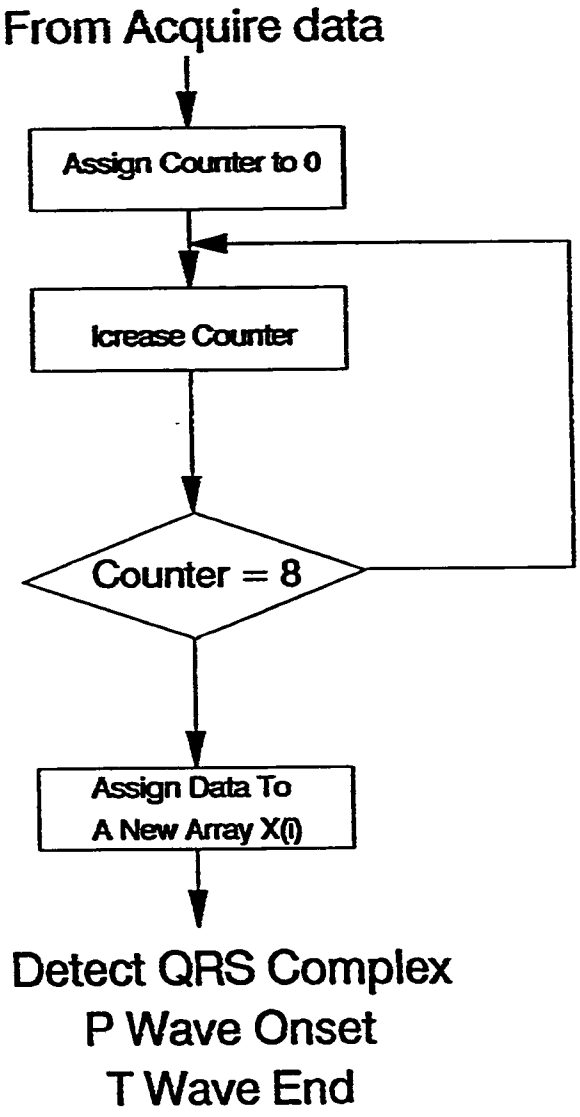


Figure 4.24: C-language Function (Minimize)

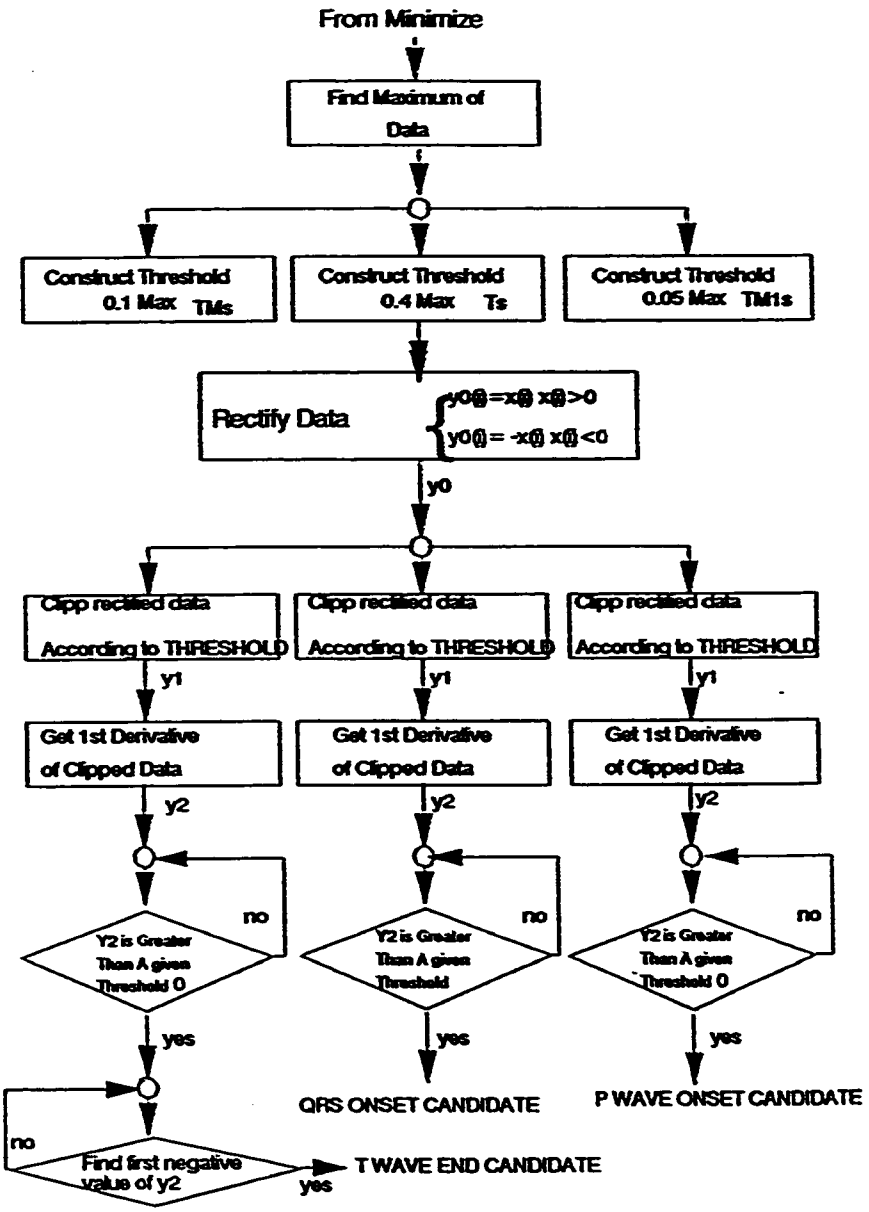


Figure 4.25: C-language Functions for the Detection of QRS,P onset and T end

THIS PAGE IS INTENTIONALLY LEFT BLANK

2. Using serial communication and no external synchronization (sy) devices are used, internal sy is used.
3. The frequency ratio of the on-chip load signal to on-chip sy signal should be faster than or equal to the sampling frequency.

In order to meet the first requirement, the bits 7 and 6 are assigned to 1,0 in order to get 16-bit integer as an input. Moreover, bits 16 and 17 are chosen to be 1, in order to assign the 16-bit integer to the upper 16 bits of input buffer . Bit 16 of the IOC is chosen to be 1 for synchronization with bit 19 although it does not affect our environment of programming [24].

To meet condition 2 bit 0,4,5 are assigned with logic 1,1,1. To meet the third requirement, bits 3,2,18 are assigned to 1,1,1. This will allow us to get synchronization frequency that will meet perfectly with the sampling frequency that is 2 KHz. because CKI=49:152 MHz, and IOC3-IOC0 = 1101 = 0xd IOC18 = 1, the internal clock generator is configured for a 2.048 MHz, i.e. 49.152 MHz / 24 bit clock. A 64 KHz load signal and a 2 KHz synchronization signal. Appendix E shows the functions of the IOC register bits.

Another important register in the DSP32C is the processor control word (PCW). The PCW is a 16-bit internal DSP32C register that can be read or written directly using DSP32C instructions. The PCW is used to configure interrupts, the PIO port user control bits, and external memory wait states. Appendix E shows the functions of the PCW register bits. As no interrupts nor parallel I/O , the bits 6-15 are cleared. However, because of the use of external memory A and B, the bits 0, 1, 3, and 5 are set to 1. Appendix C includes the DSP software listings.

According to the need of digitizing analog data fed into the input of the DSP32-C, a sampler is found on the board. In order to control the sampling frequency, a memory mapped MODE CONTROL register is found on the address 0x800000. Table 3.2 shows how to assign the sampling frequency of the device.

Table 4.2: MODE CONTROL

MODE CONTROL				(SAMPLE RATE)	
7	6	5	4	(normal mode)	(High speed mode)
0	0	0	0	32	128
0	0	0	1	16	64
0	0	1	0	8	32
0	0	1	1	4	16
0	1	0	0	2	4
1	0	0	0	100	400
1	0	0	1	50	200
1	0	1	0	25	100
1	0	1	1	12.5	50
1	1	0	0	6.25	25
0	1	0	1	22.05	88.2
0	1	1	0	44.1	176.4
0	1	1	1	external Master clock, internal FSI	
1	1	1	1	external Master clock and FSI	

4.5.3 PC Software

Building a PC software that communicates with the DSP32-C programs is the next objective. This software is to upload the necessary information from the processed signal. It also displays it on the screen. These information are either the signal itself to be displayed, or the signal features measured by the DSP32-C. The main part of the PC software runs the DSP32C program, inter-

rupts it if necessary, downloads and uploads information needed by the process. The C language program of the PC is listed in appendix D.

The first module in the PC program checks the presence of the DSP board, and lets the user allocate its address if it is not set to the factory default address (300 H). This module defines the type of microprocessor on board, memory mode, memory size and board type. It is called set_up_board().

The next module which is found in the main function interacts with the board. It uploads the information that is produced by the board by a function called dsp_up_array(addr,count,array); which uploads an array of integers from the DSP32-C memory starting at the address specified by addr, for an array size of count , and loads it in a buffer called array.

The last module is the display one. The functions plot() and plot1() set the graphics mode of the screen and display the data. Note that no processing is made by the PC. The processing power of the PC is left for any future expansion of the system to incorporate other activities.

4.5.3.1 PC Software Organization

The PC software is written in the "C" language. It represents the host that the system operator deals with. It offers the operator the capability to enter input data, display results, save the ECG and interact with the DSP-32C. In the following, the procedure implementation is described. The block diagram of the PC software is displayed in Figure 4.26.

1. Initially, the operator should be aware that he/she must specify the input channel he/she is using for data acquisition. If he/she doesn't specify the channel the system will prompt an error message and return to DOS.
2. The PC sets the DSP-32C board environment variables such as board type, memory size, memory mode and it asks the operator if the address is not set to default factory address to change it to its new settings. The system tests specific signals coming to it from that address. If it matches those that are supposed to be given to it from the DSP-32C board, it accepts the address and use it as the address of DSP-32C. If the system fails to specify the address of the board correctly, it prompts an error message and returns to DOS.

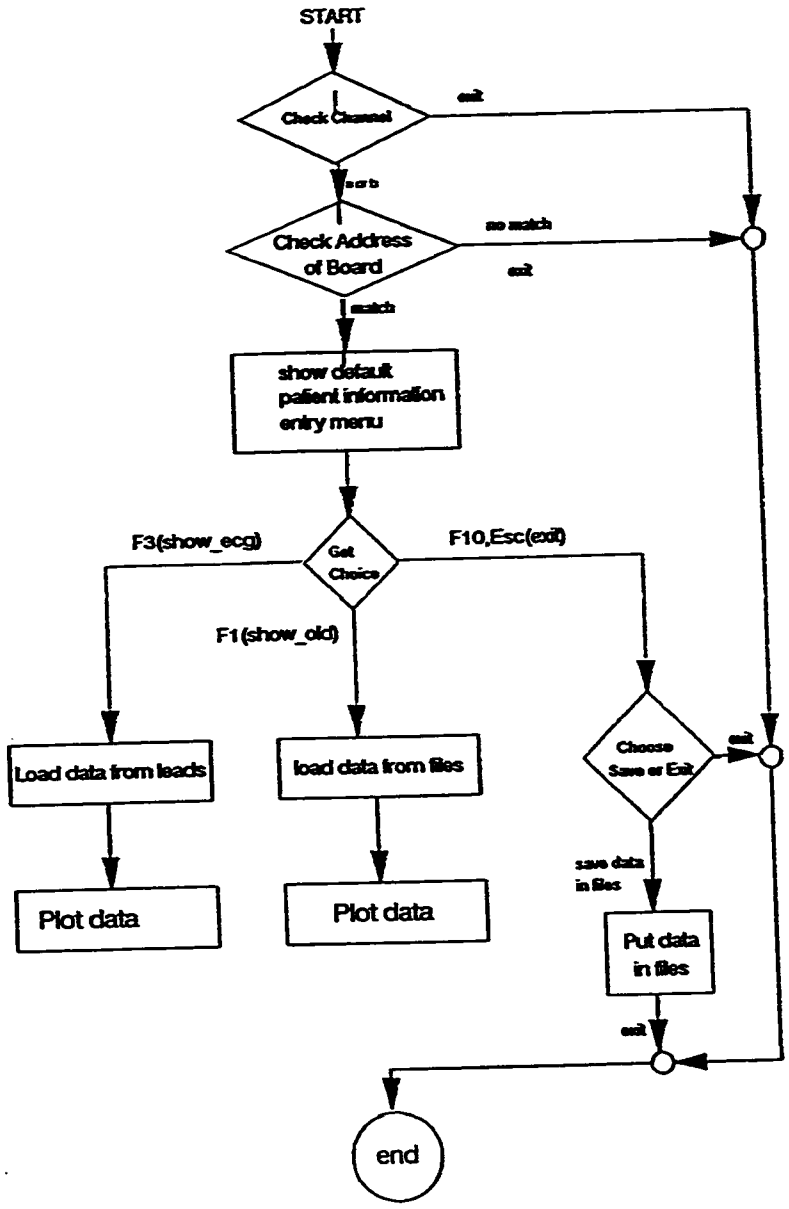


Figure 4.26: PC Software Block Diagram

3. When the system starts to run, it begins by displaying a default information data entry menu to allow the user to enter data related to patient such as name, marital status and an ID number assigned to each patient. A prompt at the bottom of the screen tells the operator what information to fill, and what format the information should be in.

4. The choice of displaying a currently recorded ECG, displaying an old ECG or to release the software is given at the bottom of information data entry menu. The data entry menu is shown in Figure 4.27.

5. If the operator selects the choice of displaying a currently recorded ECG (F3 choice), the system software draws 12 rectangles with an assigned lead to each rectangle. The system then runs the DSP-32C assembly language software which grasp the data in real time. The PC then displays each lead recording data in the corresponding rectangle after it receives the flag set from the DSP-32C program that it is done with data acquisition. After that, the PC resets the DSP-32C and then runs the DSP-32C digital signal processing "C" language software. This software is responsible for extracting the features from lead "II". The operator could change

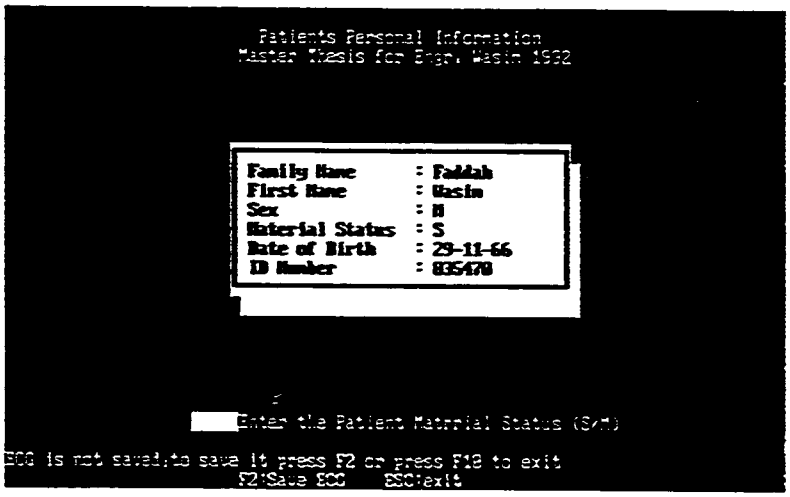


Figure 4.27: DATA ENTRY MENU OF THE SYSTEM

the input leads choice either manually or automatically.

6. The operator may display a saved ECG signal for a given ID number by pressing "F1" (show old ECG). This is performed by first asking the operator to enter the patient's ID number. If the patients ID number is not found, the system prompts an error message and returns to DOS.

7. When the operator is done with displaying a currently recorded ECG, the system asks him/her if he/she wants to save it or not, then the system

returns to DOS. However, it returns to DOS as soon as the operator is done with the display of a saved ECG.

4.6 Results

According to the work done, the ECG signals of a candidate are detected and displayed on the screen of a PC. The output for the signal aVL from our system for the candidate is shown in Figure 4.28 and an output for the same signal from an ECG machine is shown in Figure 4.29. As the two signals compared we can find that a very good output is achieved. Moreover, the system is able of detecting all the ECG readings and displaying them on the screen too. The system is capable of detecting the QT, RR and the heart beat. However, the system has a drawback in the sense of having insufficient memory for detecting the features of the signal. This is because the memory in both the DSP and the PC are not enough to deal with the huge software built for the system. The output in Figure 3.31 shows the six leads and the signal processing of lead II for the same candidate. Figure 4.30 shows each lead with its corresponding name on the top right end of the signal. So, I is printed on the top right end of

recording of lead I and so on. Figures 4.31 to 4.36 shows the output of the recordings of leads I, II, III, aVR, aVL and aVF, respectively. Each lead is recorded separately.

Another candidate who has an ECG recorded from a standard ECG machine performed an experimental ECG recording from our device. The output of the standard ECG machine is shown in Figures 4.37 to 4.42. Moreover, the output from our instrument is shown in Figures 4.43 to 4.48.

The system can also be used in monitoring the ECG signals. This means that it can also be used in ICU for monitoring patients in that unit.

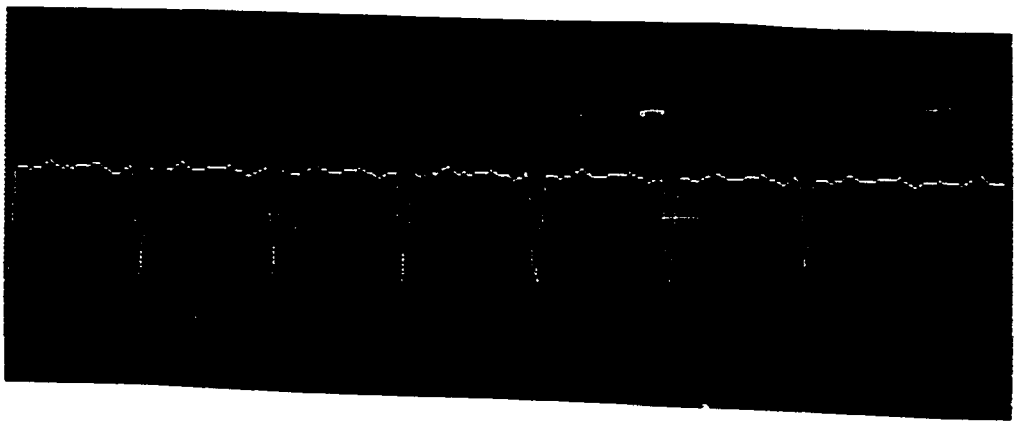


Figure 4.28: OUR SYSTEM OUTPUT

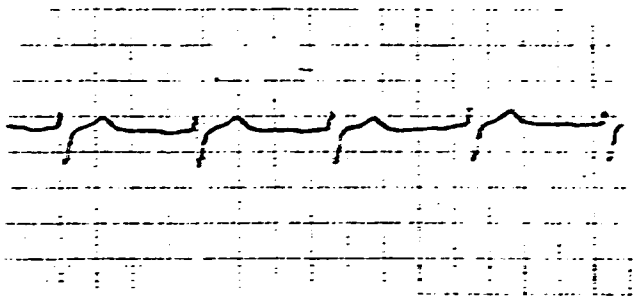


Figure 4.29: ECG MACHINE OUTPUT

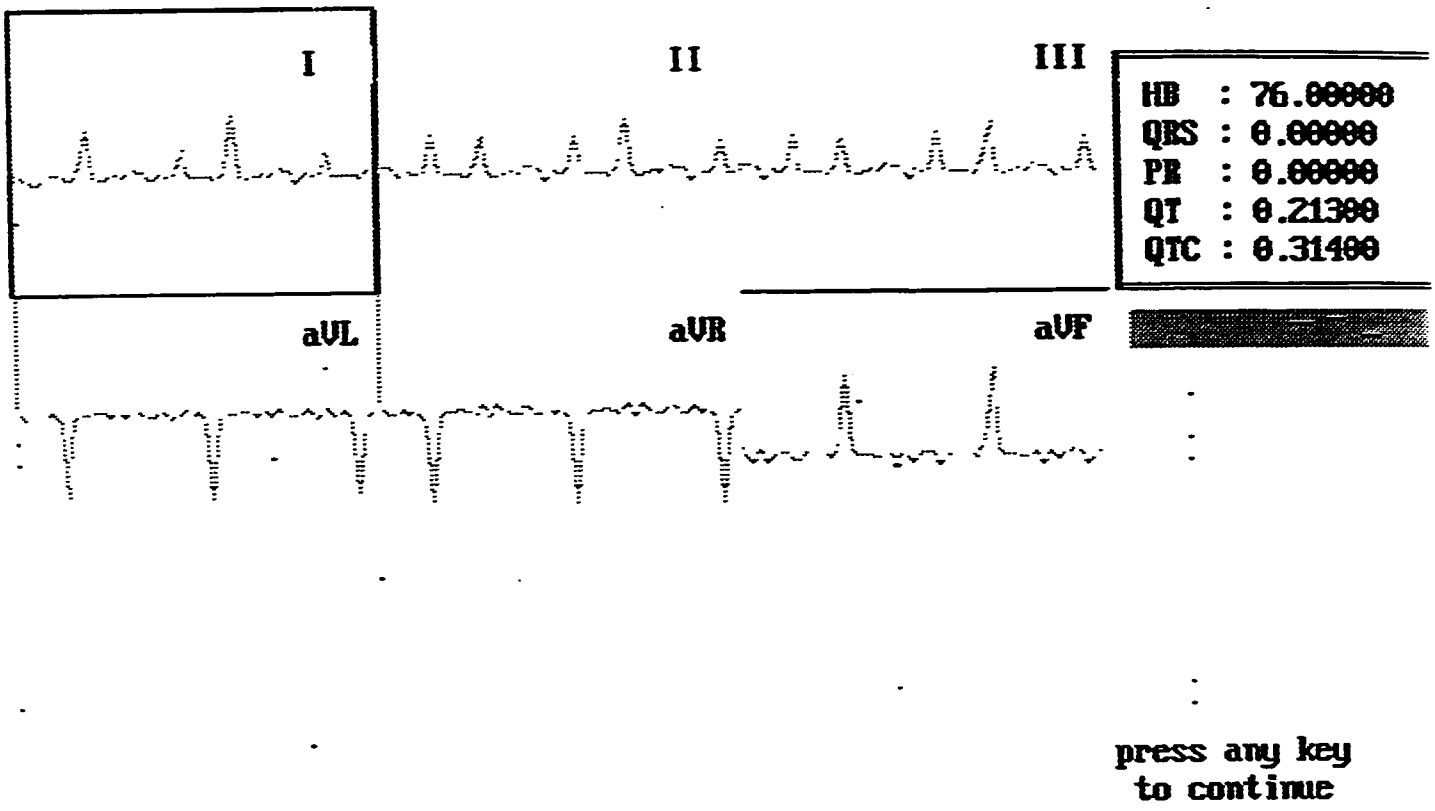


Figure 4.30: OUR SYSTEM OUTPUT FOR ALL LEADS WITH DSP

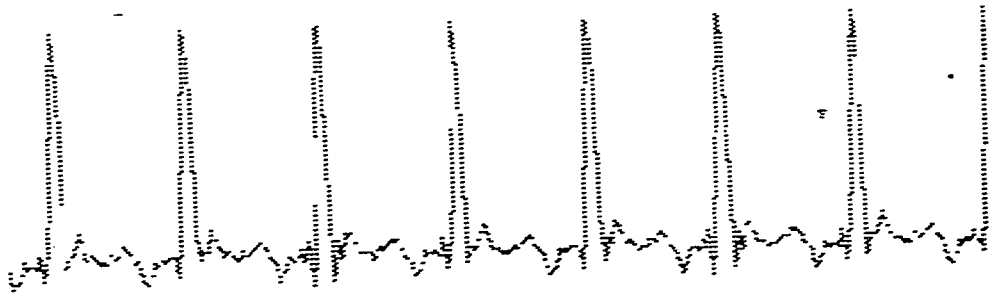


Figure 4.31: **OUR SYSTEM OUTPUT FOR LEAD I**

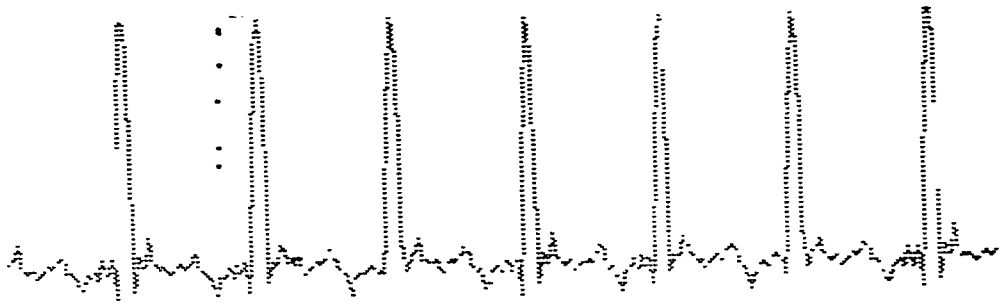


Figure 4.32: **OUR SYSTEM OUTPUT FOR LEAD II**

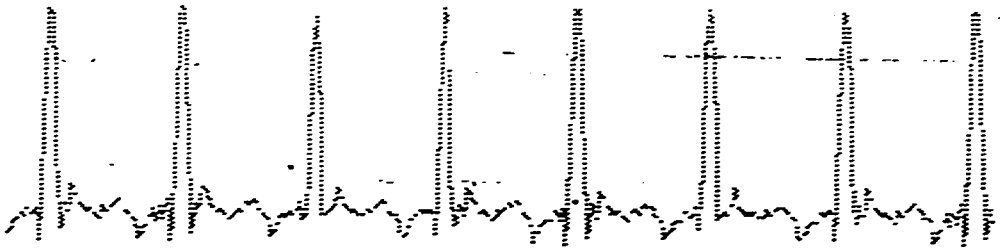


Figure 4.33: OUR SYSTEM OUTPUT FOR LEAD III

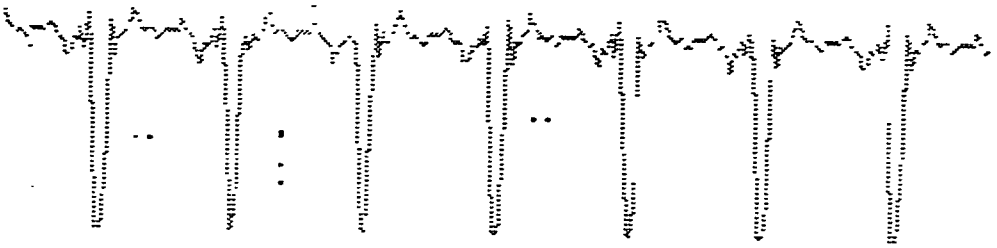


Figure 4.34: OUR SYSTEM OUTPUT FOR LEAD aVR

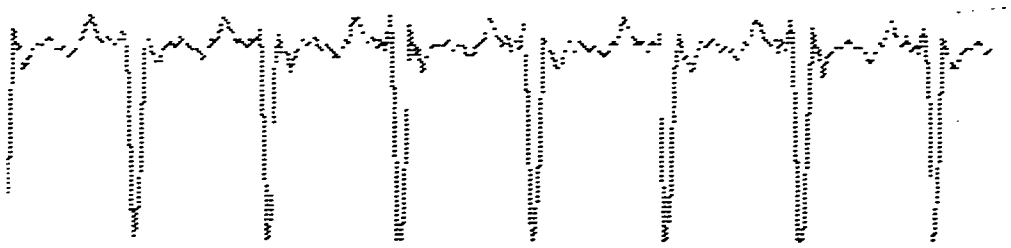


Figure 4.35: OUR SYSTEM OUTPUT FOR LEAD aVL

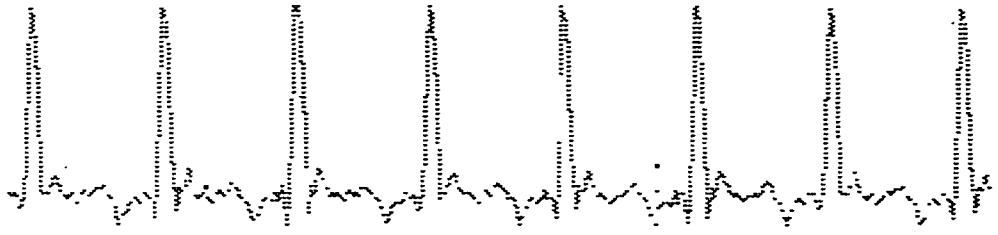


Figure 4.36: OUR SYSTEM OUTPUT FOR LEAD aVF

Figure 4.37: THE CANDIDATES STANDARD OUTPUT FOR LEAD I

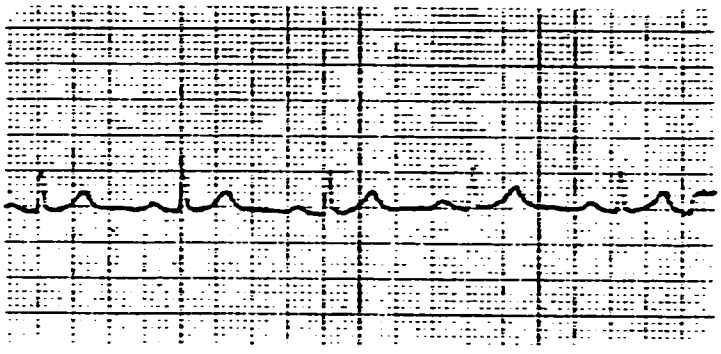


Figure 4.38: THE CANDIDATES STANDARD OUTPUT FOR LEAD II

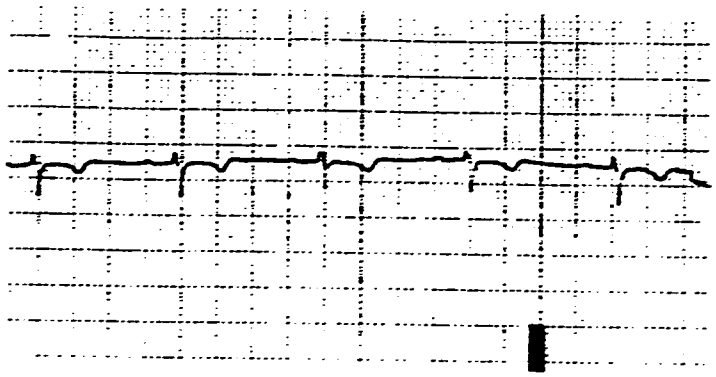


Figure 4.39: THE CANDIDATES STANDARD OUTPUT FOR LEAD III

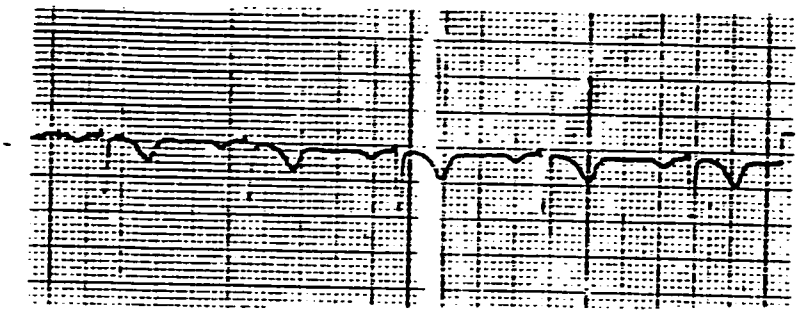


Figure 4.40: THE CANDIDATES STANDARD OUTPUT FOR LEAD aVR

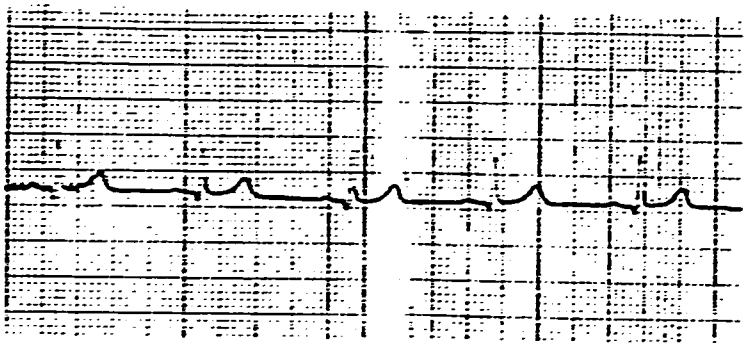


Figure 4.41: THE CANDIDATES STANDARD OUTPUT FOR LEAD aVL

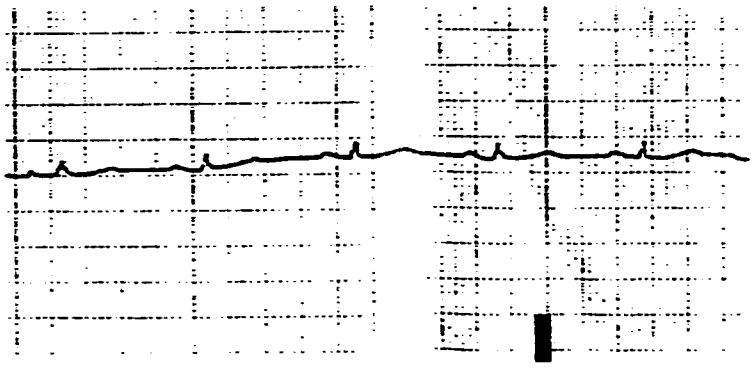


Figure 4.42: THE CANDIDATES STANDARD OUTPUT FOR LEAD aVF

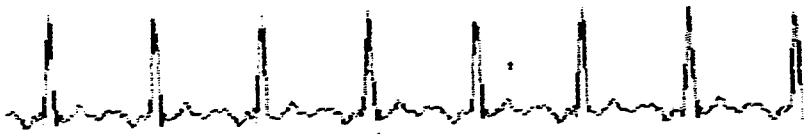


Figure 4.43: OUR SYSTEM OUTPUT FOR LEAD I FOR THE CANDIDATE

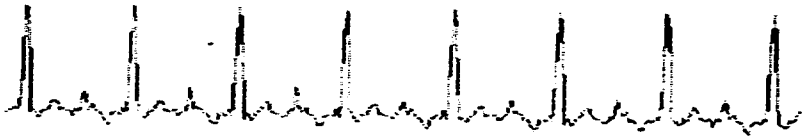


Figure 4.44: OUR SYSTEM OUTPUT FOR LEAD II FOR THE CANDIDATE

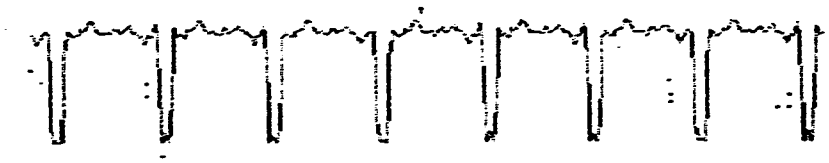


Figure 4.45: OUR SYSTEM OUTPUT FOR LEAD III FOR THE CANDIDATE

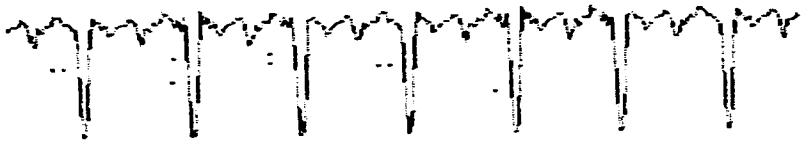


Figure 4.46: OUR SYSTEM OUTPUT FOR LEAD aVR FOR THE CANDIDATE

Figure 4.47: OUR SYSTEM OUTPUT FOR LEAD aVL FOR THE CANDIDATE

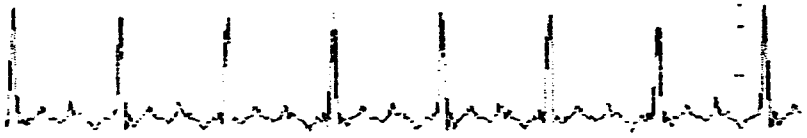


Figure 4.48: OUR SYSTEM OUTPUT FOR LEAD aVF FOR THE CANDIDATE

4.7 Performance Analysis

As a comparison between the outputs from the standard ECG machine and our device, we find that they do match in polarities and shape. This means that as the device is operating well. However, the scales are not synchronous

on the hard copies because we have used a standard built-in graphics package. With a built graphics packages, this problem of scaling could be overcome. Moreover, there are some noises that encounter with our signal. This could be explained if we notice that the connections between the different stages in the interfacing circuit are long and exposed to many interfering sources and distortions. Although the board is tested before on signals generated from signal generator and performed well, the tests are performed for different stages separately. As a whole, when the circuit is tested on the body signals, noises affected the signal according to the poor and long connections between the different stages in the circuit. Moreover, although the amplifiers used in the interfacing circuit are DG 508 which are manufactured for low voltages amplification, they are not as sensitive for the biopotential signals as some other amplifier could be. This again may be a cause for some of the additional noises found in the signal. In addition, the Notch filter built has a cut-off frequency at 59 Hz. This small deviation might be a reason for such noises. Building an exact 60 Hz Notch Filter faced the fact that the type of filter is very sensitive to the components that are used in building it. This is important to understand the fact of not being able to exactly build an exact 60 Notch Filter

due to the absence of the sensitive components (resistors, capacitors and highly-sensitive op amps) that could meet this requirement.

4.8 Summary

In this chapter the system that is built in this thesis is described. It included the description of both the hardware and the software of the system. The chapter also showed the output of the system when applied to a candidate with comparison with the output obtained from a standard ECG machine. In the next and final chapter, conclusions and proposed future study is given.

Chapter 5
CONCLUSIONS AND PROPOSED FUTURE WORK

5.1 Conclusions

In This thesis the idea of using the PC playing the role of a smart electrocardiogram (ECG) equipment is achieved. This main part of the thesis is concerned with acquiring data from the ECG leads and processing some of acquired signal. The process is done by using a digital signal processing board analog to digital converter (ADC). The input of the ADC is connected to the output of an interfacing circuit that amplifies the ECG signal that is collected by the ECG leads from the body surface. The interfacing circuit was designed in order to amplify and differentiate the different lead signals. A study about the heart as a physiological organism is performed too. Moreover, a survey of some methods of analyzing the ECG signal and extracting its features is done. An investigation of applying these methods in our study is performed in order to get the values of the QRS, PQ, QT waves and

the heart rate. Moreover, a complete study of how the digital signal processing (DSP) board called DSP32-C and its main CPU DSP32C chip is performed. C-language was used as the main programming language on this board in order to acquire and analyze the ECG signals.

Although our design is based on the AT&T DSP32C chip and the Ariel DSP board, it is important to notice that not more than 20% of the software design would be modified if another DSP board is used for the same application. The hardware, again, needs not to be modified except for the DSP board.

In our work two types of data acquisition techniques are used, namely, DMA and polled. The DMA technique is used for the data that are to be displayed. However, the polled programming is used for the data that are to be processed for feature extraction.

5.2 Recommendation for Future Work

Any of the following points could be considered as an extension to the work that is performed in this thesis:

1. Build an expert system that can detect any abnormalities from the recordings of an ECG unit. Sophisticated filing management system could be incorporated so that it can analyze the situation since the first visit paid to an M.D.

2. Build a system that detects many signals on one piece of a PC. This can be built by using the other input channel and/or the auxiliary input channel. The other signals could be EEG, EMG or EOG.
3. Build a system that controls many monitors and detects many biopotential signals simultaneously.
4. More processing of the ECG signal could be performed. This helps the M.D.'s in a more efficient diagnosis of the cases. This includes all the possible algorithms for feature extraction and/or signal simulation on PC.

Appendix A
Fortran Program to Simulate and Analyze the ECG Signal

```
C                        WRITTEN   BY
C                        WASIM FADDAH
C                      FOR A MASTER THESIS
C
C
C THIS FORTRAN PROGRAM DETERMINES THE ECG DATA FEATURES BY USING THE
C AMPLITUDE C AND FIRST DERIVATIVE METHOD. IT INCLUDES THE FOLLOWING
C PARAMETERS
C
C * X(I) WHICH IS AN ARRAY THAT INCLUDES THE DATA OF SIMULATED ECG.
C   THE ARRAY IS READ FROM AN INPUT FILE.
C * YO(I),Y1(I) AND Y2(I) WHICH ARE THE ARRAYS OF THE DATA AFTER EACH
C   STEP OF PROCESSING TO FIND QRS ONSET.
C * YOM(I), Y1M(I) AND Y2M(I) WHICH ARE THE ARRAYS OF THE PROCESSED
C   DATA FOR THE DETECTION OF THE T WAVE END.
C * YOM1(I), Y1M1(I) AND Y2M1(I) WHICH ARE THE ARRAYS OF PROCESSED DATA
C   THAT ARE USED FOR THE DETECTION OF ONSET OF P WAVE.
C * ONSET(I),ONSETM(I) AND ONSETM1(I) ARE THE ARRAYS THAT DETERMINES
C   THE DATA VALUE AT WHICH THE QRS ONSET, T END AND P ONSET ARE
C   DETERMINED RESPECTIVELY.
C * ONST(I), ONSTM(I) AND ONSTM1(I) ARE THE ARRAYS THAT CONTAIN THE
C   VALUE OF THE DISCRETE TIME AT WHICH THE QRS ONSET,T END AND P ONSET
C   OCCUR  RESPECTIVELY.
C * WG1 AND WG2 ARE WHITE GAUSSIAN NOISES ADDED TO THE SIGNAL WITH
C   WITH VARIANCES OF SD1 AND SD2 RESPECTIVELY.
C
C
      REAL X(1000),Z(1000),Y(1000),TSM(100),TSM1(100)
      REAL Y1M(1000),Y1M1(1000)
      REAL Y2M(1000),Y2M1(1000)
      REAL ONSTM(100),ONSETM(100)
      REAL ONSTM1(100),ONSE1(100)
      REAL MAXO(1000),ONST(100),SR(9),RP(9)
      REAL TS(100),ONSET(100),MAX(1000)
      REAL YO(1000),Y1(1000),Y2(1000),X1(1000),Y3(1000)
      REAL WG1,WG2,SD1,SD2
      DO 10 I=1,191
       READ(8,*)  X(I)
   10 CONTINUE
```

```
      SD1=0.1
      DO 40 K=1,191
      CALL RNNOR(1,WG1)
      CALL SSCAL(1,SD1,WG1,1)
         I=191+K
         X(I)=X(K)+WG1
   40 CONTINUE
      SD2=0.2
      DO 50 K=1,191
      CALL RNNOR(1,WG2)
      CALL SSCAL(1,SD2,WG2,1)
         I=382+K
         X(I)=X(K)+WG2
   50 CONTINUE
      SD2=0.3
      DO 51 K=1,191
      CALL RNNOR(1,WG3)
      CALL SSCAL(1,SD2,WG3,1)
         I=382+K+191
         X(I)=X(K)+WG3
   51 CONTINUE
C*****************************************************************
      NS=4
      SF=191
      SF1=SF-30
      END = NS*SF
      START=1
      DO 110 I = 1 , NS
      MAX(I)=0
C*****************************************************************
         DO 120 K=START,END
            IF (X(K) .GT. MAX(I)) THEN
               MAX(I)=X(K)
               MAXO(I)=K
            ENDIF
           DO 130 J=1,SF1
           M=K+J
               IF (M .GT. END) THEN
                  X(M)=0
               ENDIF
               IF (MAX(I) .GT. X(M)) THEN
                   GO TO 130
               ELSE
                   GO TO 120
               ENDIF
  130 CONTINUE
      TS(I) = 0.40 * MAX(I)
      TSM(I) = 0.10 * MAX(I)
      TSM1(I) = 0.05 * MAX(I)
      START = MAXO(I) + SF/2
C     WRITE (6,*) I,MAXO(I),MAX(I),TS(I),START,END
      GO TO 110
  120 CONTINUE
  110 CONTINUE
C*****************************************************************
      DO 135 K =1,END
         YO(K)=ABS(X(K))
  135 CONTINUE
```

Appendix A


```
C*********************************************************************
      DO 140 I=1,NS
         IF (I .EQ. NS) THEN
             MAXO(I+1)=END
         ENDIF
      DO 150 K=MAXO(I)-30,MAXO(I+1)-31
         IF (YO(K) .GE. TS(I))THEN
             Y1(K)=YO(K)
         ELSE
             Y1(K)=TS(I)
         ENDIF
         IF (YO(K) .GE. TSM(I))THEN
             Y1M(K)=YO(K)

         ELSE
             Y1M(K)=TSM(I)
         ENDIF
         IF (YO(K).GE.TSM1(I))THEN
             Y1M1(K)=YO(K)
         ELSE
             Y1M1(K)=TSM1(I)
         ENDIF
  150 CONTINUE
  140 CONTINUE
C*********************************************************************
      DO 160 I=1,NS
         IF (I.EQ.NS) THEN
             MAXO(I+1)=END
         ENDIF
      ONST(I) = 0
      ONSTM(I) = 0
      ONSTM1(I) = 0
          DO 170 K=MAXO(I)-29,MAXO(I+1)-29
              Y2(K) = Y1(K+1) - Y1(K-1)
C     WRITE (6,*) M ,X(M),YO(M),Y1(M),Y2(M)
             IF(Y2(K).GT.0.7)THEN
                ONST(I)=K
                GO TO 160
             ENDIF
  170 CONTINUE
  160 CONTINUE
C*********************************************************************
      COUNT=0
      DO 360 I=1,NS
         IF (I.EQ.NS) THEN
             MAXO(I+1)=END
         ENDIF
          DO 370 K=ONST(I)+29,ONST(I+1)-29
              Y2M(K) = Y1M(K+1) - Y1M(K-1)
             IF (Y2M(K).LT.0.0)THEN
               DO 111 J=1,1000
                L=K+J
               Y2M(L) = Y1M(L+1) - Y1M(L-1)
                IF(Y2M(L).EQ.0.0)THEN
                GO TO 112
                ENDIF
```

Appendix A


```
  111 CONTINUE
  112           ONSTM(I)=L
                GO TO 360
              ENDIF
  370 CONTINUE
  360 CONTINUE
C*****************************************************************
      DO 460 I=1,NS
         IF (I.EQ.NS) THEN
            MAXO(I+1)=END
         ENDIF
          DO 470 K=ONSTM(I)+29,ONSTM(I+1)-29
              Y2M1(K) = Y1M1(K+1) - Y1M1(K-1)


             IF (Y2M1(K).GT.0.0)THEN
                ONSTM1(I)=K
                GO TO 460
             ENDIF
  470 CONTINUE
  460 CONTINUE
C*****************************************************************
      DO 180 I=1,NS
          M=ONST(I)
          ONSET(I)=X(M)
          M1=ONSTM(I)
          ONSETM(I)=X(M1)
          M11=ONSTM1(I)
          ONSE1(I)=X(M11)
  180 CONTINUE
C*****************************************************************
      DO 190 I=1,NS
       IF (I .EQ. NS) THEN
              MAXO(I+1)=END
       ENDIF
         DO 200 K=MAXO(I)-30,MAXO(I+1)-31
           IF (K .EQ. ONST(I)) THEN
              WRITE (7,*) K , X(K),ONSET(I),Y1(K),Y2(K)
           ELSE
              WRITE (7,*) K , X(K),-5,Y1(K),Y2(K)
           ENDIF
  200 CONTINUE
C*****************************************************************
         DO 400 K=ONST(I)+29,ONST(I+1)+29
           IF (K.EQ.ONSTM(I)) THEN
              WRITE (9,*) K , X(K),ONSETM(I),Y1M(K),Y2M(K)
           ELSE
              WRITE (9,*) K , X(K),-5,Y1M(K),Y2M(K)
           ENDIF
  400 CONTINUE
C*****************************************************************
```

```
          DO 410 K=ONSTM(I)+20,ONSTM(I+1)+20
            IF (K.EQ.ONSTM1(I)) THEN
               WRITE (4,*) K , X(K),ONSE1(I),Y1M1(K),Y2M1(K)
            ELSE
               WRITE (4,*) K , X(K),-5,Y1M1(K),Y2M1(K)
            ENDIF
  410 CONTINUE
  190 CONTINUE
C*****************************************************************************
       DO 13 K=ONST(K)+29,ONST(K)-29
          WRITE (6,*) X(K),Y1M(K),Y2M(K)
   13  CONTINUE
       DO 165 I=1,NS-1
       SR(I)=MAXO(I+1)-MAXO(I)
       RP(I)=(SR(I)/SF)*60
          WRITE (8,*) 'THE HEART BEAT IS', RP(I)
          WRITE (8,*) MAXO(I+1),MAXO(I),SR(I),SF
  165 CONTINUE
       STOP

       END
```

APPENDIX B
C PROGRAMMING FOR DSP-32C

Written By: Wasim T. Faddah

```
/*
      ecg1mag.c
  ************************************************************************
  * This program depends on the magnitude of the ECG signal to detect*
  * the QRS complex. This happens by finding the maximum of the      *
  * signal and then set a threshold to find the onset of the QRS     *
  * complex                                                          *
  ************************************************************************
*/
/*
  ************************************************************************
  * Inclusion of the standard include header files                   *
  ************************************************************************
*/
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "dsp3addr.s"
#include "io3.asm"

#define sf 500
#define sf1 220
#define ns 4
#define end 500

float y10[2000];
float y11[2000],ym11[2000],y1m11[2000];
float y12[2000],ym12[2000],y1m12[2000];
float max[10];
float ts[10],tsm[10],tsm1[10];
float onset[10],onsetm[10],onsetm1[10];
float hb[1],pq[1],qt[1];
int   onst[10],onstm[10],onstm1[10];
int   max0[10];
float z[8000];
float x[2000];
int flagg[1];

main()
{
 int i;
 flagg[0]=0;
 get_data();
 minimize();
 find_max();
 find_abs();
 for (i = 0; i < ns ;i++)
```

```
{
 ts[i] = 0.4 * max[i];
 tsm[i] = 0.1 * max[i];
 tsm1[i] = 0.05 * max[i];
}
find_thr();
find_dsp();
dsp3t();
dsp3p();
bb[0] = 0;
pq[0] = 0;
qt[0] = 0;
  for (i = 1 ; i < ns ; i++)
       {
        bb[0] = bb[0] + onst[i] - onst[i-1];
        pq[0] = pq[0] + onst[i] - onstm1[i];
        qt[0] = qt[0] + onstm[i] - onst[i];
       }
  bb[0] = 60 / ( bb[0] / ( (ns-1) * sf ) );
  pq[0] = pq[0] / ( (ns-1) * sf  );
  qt[0] = qt[0] / ( (ns-1) * sf  );
 flagg[0]=1;
}
/* end of main */

get_data()
{
int i;
float mydata;
/* flagg[0]=0; */
set_sr(SR2);
set_ioc(0x30cc4);
       for (i=0;i<=8000;i++)
       {
         mydata = get_ibuf();
         z[i]=mydata;
       }
/* flagg[0]=1; */
}

/*
  ************************************************************************
  * This function checks for the maximum value of the ECG signal     *
  ************************************************************************
*/
minimize()
{
int counter,i,check,m;
counter=0;
i=0;
check=0;
       do
        {
        if (check == 0)
        {
         m=i/4;
         x[m] = z[i];
         counter = 0;
        }
```

```
        counter++;
        check = counter - 4;
       }while(i++ <= 8000);
}
find_max()
{
 int k,i,j;
 int istart,kstart,kend;
 istart = 0;
 kstart = 0;
       for(i = istart; i < ns ; i++ )
       {  max[i]=0;
      kend = kstart + end;
      for(k = kstart; k < kend ; k++)
      {  if ( x[k] > max[i] )
         {
          max[i] = x[k];
          max0[i] = k;
          kstart = max0[i] + sf1;
         }
      }
      }
}

/*
  ************************************************************************
  * This function finds the absolute value of the ECG signal          *
  ************************************************************************
*/
find_abs()
{
 int i;
  for ( i = 0 ; i <= 2000; i++ )
       {
        y10[i] = fabs( x[i] );
       }
}
/* end of find_abs */
/*
  ************************************************************************
  * This function constructs the threshold of the signal              *
  ************************************************************************
*/

find_thr()
{
 int i,j,k;
 int kend,kstart;
 kstart = 0;
   for (i = 0; i < 1; i++ )
   {
       kend = kstart + end;
       for (j = kstart; j <= kend ; j++)
      {
        if ( x[j] >= ts[i] )
        {
         y11[j] = x[j];
```

```
        }
        else
        {
          y11[j] = ts[i];
        }
        if ( x[j] >= tsm[i] )
        {
          ym11[j] = x[j];
        }
        else
        {
          ym11[j] = tsm[i];
        }
        if ( x[j] >= tsm1[i] )
        {
          y1m11[j] = x[j];
        }
        else
        {
          ym11[j] = tsm1[i];
        }
      }
      kstart=kend+1;
   }
}
/*
  ************************************************************************
  * This function compares  the data that calculated before with the *
  * threshold and defines the onset of a QRS complex                 *
  ************************************************************************
*/
find_dsp()
{
 int i,j,k;
 int kend,kstart;
 kstart = 0;
    for (i = 0; i < ns; i++ )
    {
        for (j = kstart; j <= kend; j++ )
       {
          y12[j] = y11[j+1] - y11[j-1];
          if ( y12[j] > 0.7 )
             {
         onset[i] = x[j];
         onst[i] = j;
             }
        }
    }
}
dsp3t()
{
 int i,j,k,l,m;
/*      loop3t: for (i=o; i < ns; i++ ) */
   for (i=0; i < ns; i++ )
/*      { for (k=onst(i)+300; k++<=onst(i+1)-300;)  */
   {
         k=onst[i]+300;
         while(k++ <= onst[i+1]-300)
          { ym12[k]=ym11[k+1]-ym11[k-1];
```

```
          if (ym12[k]<0.0)
/*        { for (j=1; j++<=1000;); */
          {j=1;
           while(j++ <= 1000)
           {l=k+j;
       ym12[l]=ym11[l+1]-ym11[l-1];
       if (ym12[l]==0.0)
/*     goto done */
       break;
           }
done:       onstm[i]=l;
          m=onstm[i];
          onsetm[i]=x[m];
/*         goto loop3t */
           break;
        }
     }
   }
}
/*end of dsp3t*/

dsp3p()
{
 int i,j,k;
/* int onstm(30),onstm1(30)
 float onse1(200),y2m1(200),y1m1(200);
 loop3p:  for (i=0; i < ns; i++) */
  for (i=0; i < ns; i++)
    { k=onstm[i]+300;
        while( k++ <= onstm[i+1]-300)
/*      { for ( k=onstm[i]+300; k++ <=onstm[i+1]-300;) */
        { y1m12[k]=y1m11[k+1]-y1m11[k-1];
          if (y1m12[k] > 0.0)
          { onstm1[i]=k;
       j=onstm1[i];
       onsetm1[i]=x[j];
/*     goto loop3p */
       break;
          }
        }
   }
}
/*end of dsp3p*/
```

APPENDIX C
C PROGRAMMING FOR PC

Written By: Wasim T. Faddah

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
/* #include <dos.h>  NEEDED TO DEFINE THE MODE USED IN OPEN */
#include <stdlib.h>
#include <graphics.h>

/* Include dsp driver header file */
#include <dsputil.h>

/*
*****************************************************************************
*                  GLOBAL VARIABLES USED FOR MENU PART                      *
*****************************************************************************
*/
#define ROW      7
#define WOR      6
#define SIZE 500
#define FALSE 0
#define TRUE 1
#define PR 0.0
#define QRS 0.0
#define pp  2

short left_boundary, right_boundary;
short upper_boundary, lower_boundary;
short height, width,maxx,maxy;
int  left,top;
float xscale,yscale,HR,QT,QTC;
int x1;
int g_driver,g_mode;

int n   = SIZE;
int yi[SIZE],yii[SIZE],yiii[SIZE];
int yavl[SIZE],yavr[SIZE],yavf[SIZE];
int yv1[SIZE],yv2[SIZE],yv3[SIZE];
int yv4[SIZE],yv5[SIZE],yv6[SIZE];
int flag,SAVE,NEG;

int choice,ch=1,chr=0;
char s[ROW][30],h[ROW][30],family[12],first[12],dofb[8],help[ROW][80];
char g[WOR][30],p[WOR][30],channel;
int i,flag=0;
char sex,status,id[8];

float hh;
int f[pp],f1[pp];
long ibuf3_addr,ibuf4_addr,ibuf,flagg_addr;
long ibuf5_addr;     /* ,ibuf4_addr,ibuf,flagg_addr; */
float flagg;
```

```
/* DSP32C addresses for global variables used in record.s */
long ibuf1_addr,flag_addr;
/*
*****************************************************************************
* THIS PROCEDURE DRAWS THE BOX FOR THE MENU.
*
*****************************************************************************
*/
draw_box()
 {
 int i;
 printf( "************************************");
 for (i=2;i<=ROW;++i)
  {
   gotoxy(1,i);
   printf("*                                  *");
  }
  gotoxy(1,i);
 printf( "************************************");
}

dr_box()
 {
 int i;
 printf( "********************");
 for (i=2;i<=WOR;++i)
  {
   gotoxy(1,i);
   printf("*                  *");
  }
  gotoxy(1,i);
 printf( "********************");
}

/*
*****************************************************************************
* THIS PROCEDURE BUILDS THE STRING OF A GIVEN FIELD INDEX 'I' BY
*
* CONCATINATING THE HEADER OF THAT FIELD WITH ITS VALUE ACCORD-
ING TO ITS TYPE*
*****************************************************************************
*/
void fill_string(int i)
 {
 if (i==1) sprintf(s[i],"%s %s",h[i],family);
 else if (i==2) sprintf(s[i],"%s %s",h[i],first);
 else if (i==3) sprintf(s[i],"%s %c",h[i],sex);
 else if (i==4) sprintf(s[i],"%s %c",h[i],status);
 else if (i==5) sprintf(s[i],"%s %s",h[i],dofb);
 else if (i==6) sprintf(s[i],"%s %s",h[i],id);
 }
```

```
/*
*************************************************************************
* THIS PROCEDURE INITIALIZE THE HEADERS OF THE FIELDS, THE FIELDS
THEMSELVS  *
* AND THE HELP AREA FOR EACH FIELD. THE FIELDS ARE LOADED WITH
THE DEFAULT  *
* VALUES                                                    *
*************************************************************************
*/
void start()
 {
  strcpy(h[1] ," Family Name      :");
  strcpy(h[2] ," First Name       :");
  strcpy(h[3] ," Sex              :");
  strcpy(h[4] ," Marital Status   :");
  strcpy(h[5] ," Date of Birth    :");
  strcpy(h[6] ," ID Number        :");
  strcpy(family ,"Faddah");
  strcpy(first ,"Wasim");
  sex='M';
  status='S';
  strcpy(dofb,"29-11-66");
  strcpy(id,"835470");
  strcpy(help[1],"Enter the Patient Family Name");
  strcpy(help[2],"Enter the Patient First Name");
  strcpy(help[3],"Enter the Patient Sex (M/F)");
  strcpy(help[4],"Enter the Patient Marital Status (S/M)");
  strcpy(help[5],"Enter the Patient Age (dd-mm-yy)");
  strcpy(help[6],"Enter the Patient ID Number");
 }

void put_inf()
 {
  strcpy(p[1] ," HB  :");
  strcpy(p[2] ," QRS :");
  strcpy(p[3] ," PR  :");
  strcpy(p[4] ," QT  :");
  strcpy(p[5] ," QTC :");
 }

/*
*************************************************************************
* THIS PROCEDURE DISPLAYS THE STRINGS S[I] FOR ALL THE FIELDS.
*
*************************************************************************
*/
void fill_box()
 {
  int i;
  for (i=1;i<ROW;++i)
   {
    gotoxy(2,i+1);
```

```
   printf(s[i]);
  }
 }
/*
*************************************************************************
* THIS PROCEDURE HILIGHTS THE FIELD WE ARE NOW ON BY CHANGING
ITS COLOR.    *
*************************************************************************
*/
void prompt(int ch)
 {
  textcolor(WHITE);
  textbackground(RED);
  window(25,8+ch,57,8+ch);
  clrscr();
  printf("%s",s[ch]);
  textcolor(RED);
  textbackground(WHITE);
  window(25,22,70,22);
  clrscr();
  printf("%s",help[ch]); /* IT ALSO PRINTS THE HELP ASSOCIATED TO THE FIELD
*/
  }
/*
*************************************************************************
* THIS PROCEDURE REMOVE THE HILIGHTING FROM THE FIELD INORDER
TO PASS IT TO  *
* THE NEXT FIELD.                                        *
*************************************************************************
*/
void unprompt(int ch)
 {
  textcolor(YELLOW);
  textbackground(BLUE);
  window(25,8+ch,57,8+ch);
  clrscr();
  printf("%s",s[ch]);
  textbackground(WHITE);
  window(20,22,70,22);
  clrscr();
  }

/*
*************************************************************************
* THIS PROCEDURE READS THE RESPONSE OF THE USER BY GETTING THE
KEYS HE IS   *
* PRESSING AND PROCESSING THEM. AT THE BEGINNING, THE HILIGHT IS
ON THE FIRST*
* FIELD. IF THE USER PRESSES DOWN_ARROW, THE HILIGHT MOVES TO THE
NEXT FIELD.*
* IF THE USER PRESSES UP_ARROW, THE HILIGHT MOVES TO THE BREVIOUS
FIELD. IF  *
* THE USER PRESSES <ENTER>, THE PRESENT FIELD IS DELETED (UNLESS HE
PRESS ESC*
```

```
* WHILE HE IS EDITING) AND THE NEW FIELD IS ACCEPTED (IT SHOULD FOL-
LOW THE  *
* FORMAT GIVEN ON THE HELP AREA). PRESSING 'S', 'L' OR 'P' CAUSES
SAVING,  *
* LOADING OR PROCESSING THE DATE RESPECTIVELY. 'Q' OR ESC ARE USED
TO EXIT.  *
*****************************************************************************
*/
int choose()
 {
  int c=0,cc=0,c1=0;
  prompt(ch);
  while (c!=27 && c!=13)
   {
    c=getch();
    if (c==0) c1=getch();
    else c1=0;
    if (cc==255)
     {
      textbackground(WHITE);
      window(4,25,77,25);
      clrscr();
      cc=0;
      prompt(ch);
     }
    if (c1==80 || c==13)    /* DOWN ARROW */
     {
      c=80;
      unprompt(ch);
       ++ch;
     }
    else if (c1==72)        /* UP ARROW */
     {
      unprompt(ch);
      --ch;
     }
    else if (c1==60)  save_data(id);   /* F2 */
    else if (c1==59)  load_data(id);  /* F1 */
    else if (c1==61)  show_ecg();      /* F3 */

    else if (c1==68) c=27;
    else if (c!=27)
     {
      chr=c;
      c=13;
     }
    if (ch<1) ch=ROW-1;
    else if (ch>ROW-1) ch=1;
    prompt(ch);
   }
  return(c);
 }
```

```
/*
*************************************************************************
* THIS PROCEDURE READS THE VALUE OF THE FIELD AND CHECKS ITS
FORMAT. TO ERASE*
* THE LAST CHARACTER ENTERED, BACKSPACE IS ALLOWED.
*
*************************************************************************
*/
void process(int *choice)
 {
 int c=0,i=2;
 char nn[12],in[12],dd[3],mm[3];
 flag=1;
 strcpy(in,"\0");
 textcolor(WHITE);
 textbackground(RED);
 window(45,8+*choice,57,8+*choice);
 clrscr();
 c=chr;
 printf("%c",c);
 while (c!=13 && c!=27 && i<=12 && c!=80 && c!=72)
  {
  if (c==8)  /* BACKSPACE */
   {
   if (i!=2)
    {
    i=i-2;
    in[strlen(in)-1]='\0';
    }
   else --i;
   clrscr();
   gotoxy(1,1);
   printf("%s",in);
   }
  else
   {
   strcpy(nn,in);
   sprintf(in,"%s%c",nn,c);
   }
   gotoxy(i++,1);
   c=getch();
   if (c==0) c=getch();
   printf("%c",c);
  }
 if (c!=27) /* DELETE THE OLD VALUE AND STORE THE NEW ONE IN THE
FIELD */
  {
  if (*choice==1) strcpy(family,in);
  else if (*choice==2) strcpy(first,in);
  else if (*choice==3 && (strcmp(in,"M")==0 ||strcmp(in,"m")==0 ||strcmp(in,"F")==0
||strcmp(in,"f")==0 )) sex=toupper(in[0]);
  else if (*choice==4 && (strcmp(in,"M")==0 ||strcmp(in,"m")==0 ||strcmp(in,"S")==0
||strcmp(in,"s")==0 )) status=toupper(in[0]);
  else if (*choice==5 )
   {
```

```
    sprintf(dd,"%c%c\0",in[0],in[1]);
    sprintf(mm,"%c%c\0",in[3],in[4]);
    if (strcmp(dd,"31")<=0 && strcmp(mm,"12")<=0 && in[2]=='-' && in[5]=='-')
      strcpy(dofb,in);
   }
  else if (*choice==6) strcpy(id,in);
  fill_string(*choice);
  unprompt(*choice);
  if (c==72)
   {
   --*choice;
   if (*choice<1) *choice=ROW-1;
   }
  else
   {
   ++*choice;
   if (*choice>ROW-1) *choice=1;
   }
  prompt(*choice);
 }
}
/* void show_inf( float HR, float QRS , float PR , float QT , float QTC) */
void show_inf()
{
 textcolor(BLACK);
 textbackground(WHITE);
 window(61,20,80,20);
 clrscr();
 printf("press any key");
 textcolor(BLACK);
 textbackground(WHITE);
 window(61,21,80,21);
 clrscr();
 printf(" to continue ");
 textcolor(YELLOW);
 textbackground(WHITE);
 window(62,3,80,3+WOR);
 clrscr();
 textcolor(BLUE);
 textbackground(WHITE);
 window(61,2,79,2+WOR);
 clrscr();
 put_inf(); /* LOAD THE DEFAULT VALUES */
 dr_box();
 for (i=1;i<WOR;++i)
 fill_inf(i);
   fill_nbox();
  while( !kbhit() );
   getch();
}
```

```
fill_inf(int i)
 {
  if (i==1) sprintf(g[i],"%s %2.5f",p[i],HR);
  if (i==2) sprintf(g[i],"%s %2.5f",p[i],QRS);
  if (i==3) sprintf(g[i],"%s %2.5f",p[i],PR);
  if (i==4) sprintf(g[i],"%s %2.5f",p[i],QT);
  if (i==5) sprintf(g[i],"%s %2.5f",p[i],QTC);
 }

fill_nbox()
 {
   int i;
   for (i=1;i<WOR;++i)
   {
     gotoxy(2,i+1);
     printf("%s",g[i]);}
 }

save_scr()
 {
  int i;
  char ff[15];
  FILE *f;
  sprintf(ff,"%s.scr",id);
  f=fopen(ff,"w");
  fprintf(f,"%s\n",family);
  fprintf(f,"%s\n",first);
  fprintf(f,"%s\n",sex);
  fprintf(f,"%s\n",status);
  fprintf(f,"%s\n",dofb);
  fprintf(f,"%s\n",id);
  fclose(f);
 }
go_scr()
 {
  restorecrtmode();
  textcolor(RED);
  textbackground(WHITE);
  window(1,1,80,24);
  clrscr();
  textcolor(BLUE);
  textbackground(WHITE);
  window(1,25,80,25);
  clrscr();
  printf("F1:Show Old ECG  F2:Save ECG  F3:Show ECG (RUN)  ESC:exit <EN-
TER>:next ");
  textcolor(YELLOW);
  textbackground(BLACK);
  window(25,9,59,9+ROW);
  clrscr();
  textcolor(WHITE);
  textbackground(BLUE);
  window(24,8,58,8+ROW);
```

```
 clrscr();
 draw_box();
 fill_box();
}

ask_id()
{
 textcolor(WHITE);
 textbackground(BLACK);
 window(1,1,80,25);
 clrscr();
 textcolor(WHITE);
 textbackground(BLUE);
 window(1,21,60,21);
 clrscr();
 printf(" Write the patients ID and then press <ENTER> ");
 gets(id);
}
show_ecg()
{
 int i,j,k,l;
 if( !dsp_dl_exec("ecgacq.lod") )
 {
   printf("Program \"ecgacq\" could not be downloaded.\n");
   exit(1);
 } /* if */
 flag_addr  = find_label_name("flag");
 printf("Flag address  0x%lx\n", flag_addr);
 ibuf1_addr = find_label_name("x");
 printf("Ibuf1 address 0x%lx\n", ibuf1_addr);

 /* Channel A data is stored in the same position as the left channel,
   and channel B is stored in the same position as the right channel. */
 if( channel == 'a' )
  channel = 'l';
 if( channel == 'b' )
  channel = 'r';
 textcolor(WHITE);
 textbackground(BLACK);
 window(1,1,80,25);
 clrscr();
 plot1();
 for (j=0;j<=1;j++)
 {
  for (k=0;k<=2;k++)
  {
      plot_init(k,j);
```

```
    sett(k,j);
    dsp_reset();
    dsp_run();

/* Upload buffer 1 and plot the data */
do
{
  flag = dsp_up_int(flag_addr);
  if( kbhit() )
  {
   getch();
   break;
  }
}while( flag == 0x1 );
switch(j)
{
case 0:
      switch(k)
      {
       case 0:
      NEG=0;
      outportb((long) 0x0214,0xa6);
      outportb((long) 0x0217,0x0);
      do{
      l=inportb((long)0x0216);
      if (kbhit()) break;
      }while( l != 0x0 );
      dsp_up_array(ibuf1_addr, (unsigned long)n, yi);
      plot( channel, n, yi , NEG);
      break;
       case 1:
      NEG=0;
      outportb((long) 0x0214,0xa6);
      outportb((long) 0x0217,0x1);
      do{
      l=inportb((long)0x0216);
      if (kbhit()) break;
      }while( l != 0x1 );
      dsp_up_array(ibuf1_addr, (unsigned long)n, yii);
      plot( channel, n, yii , NEG );
      break;
       case 2:
      NEG=0;
      outportb((long) 0x0214,0xa6);
      outportb((long) 0x0217,0x2);
      do{
      l=inportb((long)0x0216);
      if (kbhit()) break;
      }while( l != 0x2 );
      dsp_up_array(ibuf1_addr, (unsigned long)n, yiii);
      plot( channel, n, yiii , NEG );
      break;
       default:break;
       }
```

```
        break;
 case 1:
        switch(k)
        {
         case 0:
        NEG=1;
        outportb((long) 0x0214,0xa6);
        outportb((long) 0x0217,0x3);
        do{
        l=inportb((long)0x0216);
         if (kbhit()) break;
        }while( l != 0x3 );
        dsp_up_array(ibuf1_addr, (unsigned long)n, yavr);
        plot( channel, n, yavr , NEG );
        break;
         case 1:
        NEG=1;
        outportb((long) 0x0214,0xa6);
        outportb((long) 0x0217,0x4);
        do{
        l=inportb((long)0x0216);
        if (kbhit()) break;
        }while( l != 0x4 );
        dsp_up_array(ibuf1_addr, (unsigned long)n, yavl);
        plot( channel, n, yavl , NEG );
        break;
         case 2:
        NEG=0;
        outportb((long) 0x0214,0xa6);
        outportb((long) 0x0217,0x5);
        do{
        l=inportb((long)0x0216);
        if( kbhit()) break;
        }while( l != 0x5 );
        dsp_up_array(ibuf1_addr, (unsigned long)n, yavf);
        plot( channel, n, yavf , NEG );
        break;
         default:break;
         }
  }

  }
 }
outportb((long) 0x0214,0xa6);
outportb((long) 0x0217,0x1);
do{
l=inportb((long)0x0216);
if (kbhit()) break;
}while( l != 0x1 );
ddd();
show_inf();
while( !kbhit() );
getch();
```

```
  go_scr();
  cont();
}
show_old(char channel)
{
 int i,j,k;
 textcolor(WHITE);
 textbackground(BLACK);
 window(1,1,80,25);
 clrscr();
 plot1();
do
 {
 for (j=0;j<=3;j++)
 {
  for (k=0;k<=2;k++)
  {
      plot_init(k,j);
      sett(k,j);

   switch(j)
   {
   case 0:
        switch(k)
        {
         case 0:
       plot( channel, n, yi );
       break;
        case 1:
       plot( channel, n, yii );
       break;
        case 2:
       plot( channel, n, yiii );
       break;
        }
   case 1:
        switch(k)
        {
         case 0:
       plot( channel, n, yavr );
       break;
        case 1:
       plot( channel, n, yavl );
       break;
        case 2:
       plot( channel, n, yavf );
       break;
        }
   case 2:
        switch(k)
        {
         case 0:
       plot( channel, n, yv1 );
```

```
        break;
         case 1:
        plot( channel, n, yv2 );
        break;
         case 2:
        plot( channel, n, yv3 );
        break;
         }
    case 3:
         switch(k)
         {
          case 0:
        plot( channel, n, yv4 );
        break;
         case 1:
        plot( channel, n, yv5 );
        break;
         case 2:
        plot( channel, n, yv6 );
        break;
         }
     }
  }
 }
 textcolor(BLACK);
 textbackground(WHITE);
 window(61,20,80,20);
 clrscr();
 printf("press any key");
 textcolor(BLACK);
 textbackground(WHITE);
 window(61,21,80,21);
 clrscr();
 printf(" to continue ");
 getch();
while( !kbhit() );
}while( !kbhit() );
getch();
}

save_data( char id[])
{
 char ff[15];
 int i;
 FILE *f;
 save_scr();
 sprintf(ff,"%s.dat",id);
 f=fopen(ff,"w");
 if (channel=='r')
   {
    fprintf(f,"%c\n",channel);
    for(i=2;i<n;i+=2)  fprintf(f,"%d\n",yi[i]);
    for(i=2;i<n;i+=2)  fprintf(f,"%d\n",yii[i]);
    for(i=2;i<n;i+=2)  fprintf(f,"%d\n",yiii[i]);
```

```
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yavr[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yavl[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yavf[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yv1[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yv2[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yv3[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yv4[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yv5[i]);
    for(i=2;i<n;i+=2) fprintf(f,"%d\n",yv6[i]);
   }
  if (channel=='I')
   {
    fprintf(f,"%c\n",channel);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yi[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yii[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yiii[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yavr[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yavl[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yavf[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yv1[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yv2[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yv3[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yv4[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yv5[i]);
    for(i=3;i<n;i+=2) fprintf(f,"%d\n",yv6[i]);
   }
  fclose(f);
  SAVE=1;
}


load_data( char id[] )
{
 char ff[15],s[15];
 int i;
 FILE *f;
 ask_id();
 sprintf(ff,"%s.scr",id);
 f=fopen(ff,"r+");
 if(f==NULL)
 {
  fclose(f);
  textcolor(WHITE);
  textbackground(BLACK);
  window(20,21,60,21);
  clrscr();
  printf("Can not find the id %s",id);
 }
 else
 {
```

```
fscanf(f,"%s\n",family);
fscanf(f,"%s\n",first);
fscanf(f,"%s\n",sex);
fscanf(f,"%s\n",status);
fscanf(f,"%s\n",dofb);
fscanf(f,"%s\n",id);
fclose(f);
clear();
 textcolor(YELLOW);
 textbackground(BLUE);
 window(2,1,32,1);
 clrscr();
 printf("Press <ENTER> to see Old ECG");
 while( getch() != 13 );
sprintf(ff,"%s.dat",id);
f=fopen(ff,"r");
fscanf(f,"%c\n",channel);
if (channel=='r')
  {
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yi[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yii[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yiii[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yavr[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yavl[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yavf[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yv1[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yv2[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yv3[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yv4[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yv5[i]);
   for(i=2;i<n;i+=2)  fscanf(f,"%d\n",yv6[i]);
  }
if (channel=='l')
  {
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yi[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yii[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yiii[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yavr[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yavl[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yavf[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yv1[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yv2[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yv3[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yv4[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yv5[i]);
   for(i=3;i<n;i+=2)  fscanf(f,"%d\n",yv6[i]);
  }
fclose(f);
show_old(channel);
}
```

```
go_scr();
 cont();
}
cont()
{
choice=choose();
while(choice!=27)
{
 process(&ch);
 choice=choose();
}
}
/* Subroutine to set up PC32C/DSP32C board.
  The subroutine checks if the DSP32 environment variable is defined.
  If DSP32 is not defined, the program uses 0x300 as the base address
  of the card, or else it tries to find the board at the defined address.
*/
set_up_board()
{
 unsigned int base_address;   /* base address of board */
 char ch;
 char *environ_32addr; /* char pointer to DSP32 environment variable string */
 char temp_string[20]; /* char string to hold environment variable string */
            /* you cannot change the string pointed to by environ_32addr */
 /* Check if the environment variable is set */
 if( (environ_32addr = getenv("DSP32") )  == NULL )
 {
   /* if it is not set, ask the user if they want to use an
      address other than the default address 0x300. */
   printf("DSP32 environment variable has not been set.\n");
   printf("Default address 0x300 will be used.\n");
   printf("Do you want to change default address ? [y/n] ");
   do
   {
     ch = getch();
     ch = toupper(ch);
   } while ( ch != 'Y' && ch != 'N' );
 } /* if */
 else
 {  /* if it is set, use it. */
   strcpy(temp_string, environ_32addr);
   temp_string[3] = '\0';
   ch = ' ';
 }
 putch( ch );
 printf("\n");
 if( ch == 'Y' )
 {
  /* To use an address other than specified by the DSP32 environment
     variable, the following steps have to be taken.  Specify a
     base address, set the global variable device_type equal to 6
     and call find_device_type().  If the device is found, the program
     will return to the calling program after which you should call
     dsp_board_addr() to set all the addresses for that board.  You may
```

```
    use this technique to switch between boards in a multi-processor
    environment.
  */
  printf("\nEnter base address (in hex): 0x");
  scanf("%x", &base_address );
  board_type = 6;
  if ( find_device_type( base_address ) == 0 )
   exit(1);
  dsp_board_addr( base_address );
 }
 else
 {
  base_address = default_addr();
  if( base_address == 0 )
   exit(1);
 }
}
/* set_up_board() */

plot(char channel, int n, int y[], int NEG)
{
  int ymax;
  ymax = 0x5FFF;
  yscale = -(float)height / (float)ymax;
  yscale /= 2.;
  xscale = (float) width / (float) n;
  setviewport( left_boundary, upper_boundary, right_boundary, lower_boundary,1);
  setcolor(YELLOW);
  setbkcolor(BLACK);
 setviewport( left_boundary+1, upper_boundary+1, right_boundary+1, lower_bounda-
ry+1,1);
  if (NEG == 1)
  {
  if (channel == 'r' )
  {
      moveto( 0 , (int)  ( y[0] * yscale +height/2-50) );

      for( x1 = 2; x1 < n; x1+=10 )
       lineto( (int) ( x1 * xscale ), (int) ( - y[x1] * yscale +height/2-50) );
  }
  else if ( channel == 'l' )
  {
      moveto( 0 , (int) ( y[1] * yscale +height/2-50) );
      for ( x1 = 3; x1 < n; x1+=10 )
       lineto( (int) ( x1 * xscale ), (int) ( - y[x1] * yscale +height/2-50) );
  }
  }
```

```
else if(NEG == 0)
  {
    if (channel == 'r' )
  {
       moveto( 0 , (int) ( y[0] * yscale +height/2+50) );
       for( x1 = 2; x1 < n; x1+=10 )
        lineto( (int) ( x1 * xscale ), (int) ( y[x1] * yscale +height/2+50) );
  }
  else if ( channel == 'I' )
  {
       moveto( 0 , (int) ( y[1] * yscale +height/2+50) );
       for ( x1 = 3; x1 < n; x1+=10 )
        lineto( (int) ( x1 * xscale ), (int) ( y[x1] * yscale +height/2+50) );
  }
  }
  return( TRUE );
}

plot10
{
       g_driver=VGA;
       g_mode=VGAHI;
       initgraph(&g_driver,&g_mode,"");
       if (g_driver<0)
       {
        printf("No Driver is available");
        exit(0);
       }
}
sett(left,top)
{
char buffer;
switch (top)
{
case 0:
       switch (left)
       {
       case 0:
         textcolor(BLACK);
         textbackground(WHITE);
         window(17,2,20,2);
         clrscr();
         printf("I");
       break;
       case 1:
         textcolor(BLACK);
         textbackground(WHITE);
         window(37,2,40,2);
         clrscr();
         printf("II");
```

```
        break;
        case 2:
          textcolor(BLACK);
          textbackground(WHITE);
          window(57,2,60,2);
          clrscr();
          printf("III");
        break;
        }
        break;
case 1:
        switch (left)
        {
        case 0:
          textcolor(BLACK);
          textbackground(WHITE);
          window(17,9,20,9);
          clrscr();
          printf("aVL");
        break;
        case 1:
          textcolor(BLACK);
          textbackground(WHITE);
          window(37,9,40,9);
          clrscr();
          printf("aVR");
        break;
        case 2:
          textcolor(BLACK);
          textbackground(WHITE);
          window(57,9,60,9);
          clrscr();
          printf("aVF");
        break;
        }
        break;
case 2:
        switch (left)
        {
        case 0:
          textcolor(BLACK);
          textbackground(WHITE);
          window(17,16,20,16);
          clrscr();
          printf("V1");
        break;
        case 1:
          textcolor(BLACK);
          textbackground(WHITE);
          window(37,16,40,16);
          clrscr();
          printf("V2");
```

```
        break;
        case 2:
          textcolor(BLACK);
          textbackground(WHITE);
          window(57,16,60,16);
          clrscr();
          printf("V3");
        break;
        }
        break;
case 3:
        switch (left)
        {
        case 0:
          textcolor(BLACK);
          textbackground(WHITE);
          window(17,24,20,24);
          clrscr();
          printf("V4");
        break;
        case 1:
          textcolor(BLACK);
          textbackground(WHITE);
          window(37,24,40,24);
          clrscr();
          printf("V5");
        break;
        case 2:
          textcolor(BLACK);
          textbackground(WHITE);
          window(57,24,60,24);
          clrscr();
          printf("V6");
        break;
        }
        break;
default:
        break;
}
}
clear()
{
closegraph();
restorecrtmode();
textcolor(RED);
 textbackground(WHITE);
 window(1,1,80,24);
 clrscr();
 textcolor(BLUE);
 textbackground(WHITE);
 window(1,25,80,25);
 clrscr();
```

```
 printf("  F1:Show Old ECG     F3:Show ECG (RUN)     F10,ESC:exit <EN-
TER>:next ");
 textcolor(BLACK);
 textbackground(WHITE);
 window(4,2,76,2);
 clrscr();
 printf("                  Patients Personal Information ");
 window(24,3,58,3);
 clrscr();
 printf(" Master Thesis for Engr. Wasim 1992");
 textcolor(YELLOW);
 textbackground(BLACK);
 window(25,9,59,9+ROW);
 clrscr();
 textbackground(BLUE);
 window(24,8,58,8+ROW);
 clrscr();
 draw_box();
 for (i=1;i<ROW;++i)
  fill_string(i);
 fill_box(); /* DISPALY THE FIELDS */
}
plot_init(left,top)
{
 maxx = getmaxx();
 maxy = getmaxy();
 maxx = maxx / 4;
 maxy = maxy / 4 - 1 ;
 left_boundary = left*maxx;
 right_boundary = (left+1)*maxx;
 upper_boundary = top*maxy;
 lower_boundary = (top+1)*maxy;
 height = abs((int) (upper_boundary - lower_boundary));
 width = right_boundary - left_boundary;
 setcolor(GREEN); /* green */
  /*
  set clip region
  */
/* setviewport( left_boundary, upper_boundary, right_boundary, lower_boundary,1);*/
 rectangle(left_boundary,upper_boundary,right_boundary,lower_boundary);

  return( TRUE );
}

ddd()
{
 int i,j,ymax;
 int hb[6],qt[6],hhh,qq;
 float HHH;
 ymax = 0x7FFF;
```

```
 if ( !dsp_dl_exec("www") )
 {
      printf("help\n");
      exit(1);
 }
 ibuf4_addr = find_label_name("max");
 ibuf3_addr = find_label_name("onst");
 ibuf5_addr = find_label_name("onstm");
 flag_addr = find_label_name("flagg");
dsp_reset();
dsp_run();
        do{
           flagg = dsp_up_int(flagg_addr);
      if ( kbhit() )
      { getch();
       break;}
         }while( flagg == 0 );
      dsp_up_array(ibuf3_addr, (unsigned long) pp, f);
      dsp_up_array(ibuf5_addr, (unsigned long) pp, f1);
      hh=dsp_up_fp(ibuf4_addr);
      hh = hh / (float) ymax;
      hhh=0;
      qq=0;
      for ( i=0; i<=pp ;i++)
         {
         hhh=hhh + f[i+1] - f[i];
         qq=qq + f1[i] - f[i];
         }
         if (hhh != 0 )
         {
         HR= 250/hhh*60;
         HHH=hhh/250;
         }
         QT= qq/(3*250);
         if (QT >= 0 && hhh > 0 )  QTC= QT * sqrt( 1000 / HHH);
}


/*
*****************************************************************************
*                          THE MAIN PROGRAM                                 *
*****************************************************************************
*/
main(argc,argv)
int argc;
char *argv[];
 {
 int c=0,choi;
 argc=2;
 if( argc != 2 )
```

```
 {
  printf("Usage: record [{l,a},{r,b}]\n");
  printf("Microphone: l for left channel, r for right channel.\n");
  printf("DSP-32C:    a for Channel A, b for Channel B.\n");
  exit(1);
 }
/* channel = *argv[1];*/
 channel='a';
 if( channel != 'r' && channel != 'l' && channel != 'a' && channel != 'b' )
 {
  printf("Usage: record [{l,a},{r,b}]\n");
  printf("Microphone: l for left channel, r for right channel.\n");
  printf("DSP-32C:    a for Channel A, b for Channel B.\n");
  exit(1);
 } /* if */
 set_up_board();
 textcolor(RED);
 textbackground(WHITE);
 window(1,1,80,24);
 clrscr();
 textcolor(BLUE);
 textbackground(WHITE);
 window(1,25,80,25);
 clrscr();
 printf("  F1:Show Old ECG     F3:Show ECG (RUN)     F10,ESC:exit <EN-
TER>:next ");
 textcolor(BLACK);
 textbackground(WHITE);
 window(4,2,76,2);
 clrscr();
 printf("                    Patients Personal Information ");
 window(24,3,58,3);
 clrscr();
 printf(" Master Thesis for Engr. Wasim 1992");
 textcolor(YELLOW);
 textbackground(BLACK);
 window(25,9,59,9+ROW);
 clrscr();
 textbackground(BLUE);
 window(24,8,58,8+ROW);
 clrscr();
 start(); /* LOAD THE DEFAULT VALUES */
 draw_box();
 for (i=1;i<ROW;++i)
  fill_string(i);
 fill_box(); /* DISPALY THE FIELDS */
 choice=choose();
 while (choice!=27)
  {
   process(&ch);
   choice=choose();
  }
 textcolor(BLUE);
```

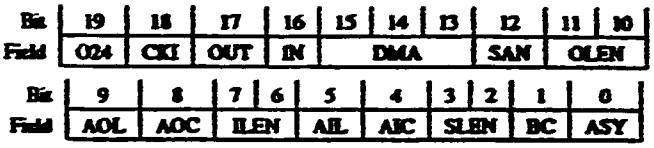
```
textbackground(WHITE);
window(1,25,80,25);
clrscr();
printf("                    F2:Save ECG    F10:exit  ");
choi=getch();
if (SAVE != 1)
{
do
{
  while (choi==27)
  {
    textcolor(BLACK);
    textbackground(WHITE);
    window(1,24,79,24);
    clrscr();
    printf("          To save ECG press F2 or press F10 to exit");
    choi = getch();
  }
  if (choi == 0)
  {
    c=getch();
       if (c==60) save_data(id);
    textcolor(BLACK);
    textbackground(WHITE);
    window(1,24,79,24);
    clrscr();
    printf("                ECG is saved...  press F10 to exit");
       if (c==68) break;
  }
}while(choi != 27 || (choi != 0 && c != 60) || (choi != 0 && c !=68));
}
textcolor(WHITE);
textbackground(BLACK);
window(1,1,80,25);
clrscr();
}
```

Appendix D
PCW for DSP32C

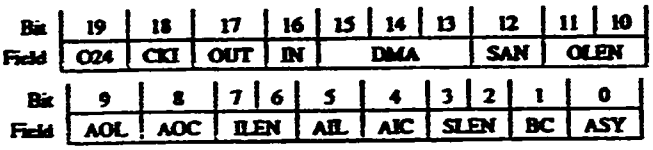
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	INTER								PIOPH	PIOPL	MEMA		MEMB		WA	WB

Bit(s)	Field	Function
0	WB	If 0, disable wait-state generator for external memory partition B. If 1, enable wait-state generator for external memory partition B.
1	WA	If 0, disable wait-state generator for external memory partition A. If 1, enable wait-state generator for external memory partition A.
3,2	MEMB	These bits select the number of wait states that are generated for the external memory in partition B. 00 — 1 wait state 01 — 2 wait states 10 — 3 wait states 11 — 2-or-more wait states (controlled by SRDYN signal)
5,4	MEMA	These bits select the number of wait states that are generated for the external memory in partition A. 00 — 1 wait state 01 — 2 wait states 10 — 3 wait states 11 — 2-or-more wait states (controlled by SRDYN signal)
6	PIOPL	If 0, PIOP[0—3] are inputs. If 1, PIOP[0—3] are outputs.
7	PIOPH	If 0, PIOP[4—7] are inputs. If 1, PIOP[4—7] are outputs.
15—8	INTER	Each bit in this field corresponds to one of the eight sources for an interrupt. A value of 1 in a position enables the corresponding interrupt source; a value of 0 disables the source. Bit Interrupt Source 8 — RESERVED. 9 — RESERVED. 10 — INTREQ2, Interrupt 2 pin 11 — OBE. Serial output buffer empty 12 — IBF. Serial input buffer full 13 — PDE. Parallel data empty (output) 14 — PDF. Parallel data full (input) 15 — INTREQ1. Interrupt 1 pin

Appendix E
IOC register for DSP32C



Bit(s)	Field	Function
0	ASY	If 0, SY is external. If 1, SY is internal. When generated internally, SY = {ICK, OCK}/{256, 512, 1024} = IOC[BC]/(32×IOC[SLEN]).
1	BC	If 0, ICK is used to derive the on-chip load and SY clocks. If 1, OCK is used to derive the on-chip load and SY clocks.
3,2	SLEN	These bits select the frequency ratio of the on-chip load clock to the on-chip SY clock. Following are the possible settings. 00 — not used 01 — Ratio = 8 10 — Ratio = 16 11 — Ratio = 32
4	AIC	If 0, ICK is external. If 1, ICK is generated internally with a frequency of CKI/8 or CKI/24, based on IOC[18].
5	AIL	If 0, ILD is external. If 1, ILD is generated internally with a frequency of {ICK, OCK}/32 = IOC[BC]/32.
7,6	ILEN	These bits specify the length of the input serial data. 00 — Serial input length is 32 bits (before ILD) 01 — Serial input length is 8 bits (after ILD) 10 — Serial input length is 16 bits (after ILD) 11 — Serial input length is 32 bits (after ILD)
8	AOC	If 0, OCK is external. If 1, OCK is internally generated with a frequency of CKI/8 or CKI/24, based on IOC[18].
9	AOL	If 0, OLD is external. If 1, OLD is internally generated with a value of {ICK, OCK}/32 = IOC[BC]/32.
11,10	OLEN	These bits specify the length of the serial output data. 00 — Used with bit 19 (O24)* 01 — Serial output length is 8 bits 10 — Serial output length is 16 bits 11 — Serial output length is 32 bits
12	SAN	If 0, clear sanity. If 1, set sanity.

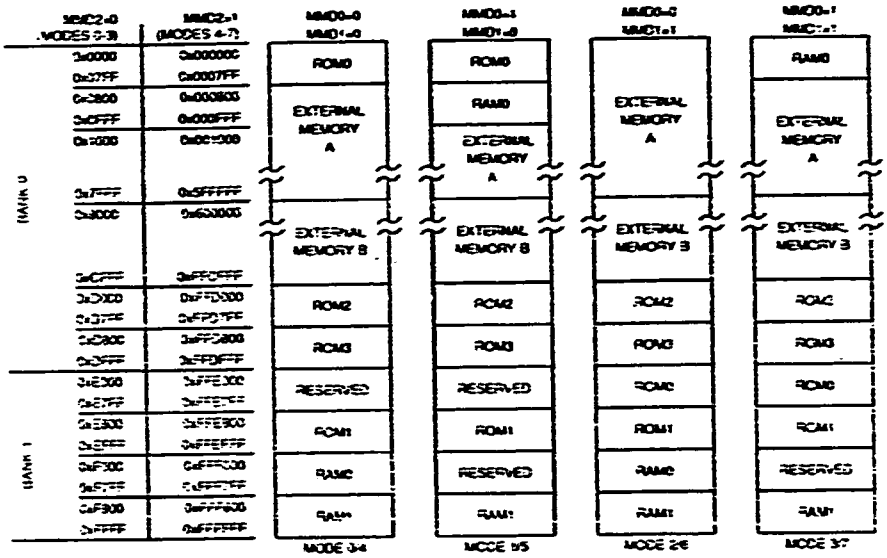


Bit	19	18	17	16	15	14	13	12	11	10
Field	O24	CKI	OUT	IN	DMA			SAN	OLEN	

Bit	9	8	7	6	5	4	3	2	1	0
Field	AOL	AOC	ILEN		AIL	AIC	SLEN		BC	ASY

Bit(s)	Field	Function
15—13	DMA	These bits control direct memory accesses. 000 — no DMA 001 — input DMA when IBF is high 010 — output DMA when OBE is high 011 — input DMA when IBF is high and output DMA when OBE is high 101 — input and output DMA when IBF is high 110 — input and output DMA when OBE is high 111 — input and output DMA when either IBF or OBE is high
16	IN	If 0, the LSB is received first during serial inputs. If 1, the MSB is received first during serial inputs.
17	OUT	If 0, the LSB is transmitted first during serial outputs. If 1, the MSB is transmitted first during serial outputs.
18	CKI	If 0, the internal SIO clock has a value of CKI/8. If 1, the internal SIO clock has a value of CKI/24.
19	O24	If 0, the output length is specified by the OLEN field of the IOC. If 1, and OLEN=0 and OUT=0 (LSB first), the output length is 24 bits.

Appendix F
DSP32C Memory Mapping



DSP32C Memory Mode Configurations (with on-chip ROM)

Appendix F

The following is the memory mapping used by the program in this thesis:

The data is loaded in the memory section *.extA,* while the text is loaded in the memory section *.extB,* according to the following memory map to the memory.

```
MEMORY {
        .mextA: o=0x0 , l=0x100000
        .mram0: o=0xfff000 , l=0x800
        .mram1: o=0xfff800 , l=0x800
        .mextB: o=0x600000 , l=0x100000
        .mram2: o=0xffe000 , l=0x800
}

SECTIONS {
        .ram0:          {} > .mram0
        .ram1:          {} > .mram1
        .extA:          {} > .mextA
        .extB:          {} > .mextB
        .ram2:          {} > .mram2
}
```

REFERENCES

1. Bronzino J., *"Biomedical Engineering and Instrumentation, Basic Concept and Application,"* N.J.:Prentice Hall Inc. 1982.

2. Webster J. et al, *"Medical Instrumentation : Application and Design ,"* Houghton Mifflin Company 1978.

3. Tompkins W., Webster J., *"Design of Microcomputer Based Medical Instrumentation,"* N.J.: Prantice Hall International 1981.

4. DeMarre D.A., Micheels D., *"Bioelectronic Measurements ,"* N.J.:Prentice Hall Inc. 1982.

5. Kember N. F., *"An Introduction to Computer Applications in Medicine,"* London: Edward Arnold ,1982.

6. *"Manual of ECG ,"* Department of Cardiology Servier Laboratories.

7. Geddes L.A., Baker L.E. *"Principles of Applied Biomedical Instrumentation ,"* N.Y.: Wiley-interscience Publication ,1975.

8. Friesen G.M. et al, *"A Comparison of the Noise Sensitivity of Nine QRS Detection Algorithms,"* IEEE Trans. Biomed. Eng., vol 37, no.1, pp.85-98,1990.

9. Fraden J. , Neuman M.R. *"QRS Wave Detection ,"* Med. Biol. Eng. Comput., Vol. 18, pp. 125-132,1980.

10. Ahlstrom M. L. and Tompkins W.J., *"Automated High-Speed analysis of Holter types with Microcomputers,"* IEEE Trans. Biomed. Eng., vol BME-30,pp.651-657 Oct 1983.

11. Okada M., *"A Digital Filter for the QRS Complex Detection ,"* IEEE Trans. Biomed. Eng. vol. BME-26, pp. 700-703, Dec 1979.

12. Parsons W.T., *"Voice and Speech Processing,"* N.Y.: McGraw Hills, 1987.

13. Lin K., Chang W. H., *"QRS Feature Extraction Using Linear Prediction ,"* IEEE Trans. Biomed. Eng., vol 36, No. 10 ,pp. 1050-1055, Oct 1989.

14. Le-Huy P. , Yvroud E., and Dion ,J.L., *"A versatile Cardiac Arrhythmia Simulator,"* IEEE Trans. Inst. Meas.,V IM_36, n 4, Dec 1987.

15. Sheridan, D.J. et al, *"A Microcomputer System for Real-Time Analysis of Cardiac Action Potential,"* Journal of Medical Engineering and Technology, V 7, n 5, sept.-oct. 1983, pp 238-242.

16. Lehner R. and Rangayyan R. M., *"A Three-Channel Microcomputer System for Segmentation and Characterization of the Phonocardiogram,"* IEEE Trans. Biomed. Eng., V BME-34, n 6, June 1987, pp 485-489.

17. Tompkins W., Shuqian W. J., *"Microcomputer Based Simulator of a 12-lead electrocardiogram,"* Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology, 1988, Nov 4-7.

18. Nasipori M., Kundu M., Basu D. K. *"Microcomputer-Based Simultaneous Processing of Multiple Holter Tapes,"* 4th IEEE Region 10th International 1989 Nov 22-24.

19. Jossinet J. et al, *"Computerized Bioelectrical Cardiac Monitor,"* Computers in Biology and Medicine,V 20, n 4, 1990 p 253-260.

20. Lucia F. D. et al, *"Microcomputer-Based Coronary Care Unit Central Station,"* Journal of Clinical Engineering, V 15, n 5, 1990 p 381-389.

21. Perkusich et al, *"Expert ECG Acquisition and Analysis System,"* Computers in Biology and Medicine,V 20, n 4, 1990 p 253-260.

22. *"WE DSP32 and DSP32C, C-language Compiler, Development System User Manual ,"* AT&T 1988.

23. *"WE DSP32 and DSP32C, C-language Compiler, Library Reference Manual ,"* AT&T 1988.

24. *"WE DSP32C, Digital Signal Processor Information Manual ,"* AT&T 1988.

25. *"Reference Manual for ECG ,"* Fukuda 1990.

26. Howard M., *"The Design of active Filters, with experiments,"* Buggbooks ,1979.

27. *"WE DSP32 and DSP32C, Support Software Library ,"* AT&T 1988.

28. Hargert, *"The ABC of Turbo C ,"* Singapore: Tech publication ,1988.

29. *"Microsoft C Compiler, Code View and the C language Reference Manual,"* Microsoft Corporation , 1986.

30. *"Microsoft C Compiler, Run time Library, Reference Manual,"* Microsoft Corporation , 1986.

31. *"Microsoft C Compiler, User's Guide, Reference Manual,"* Microsoft Corporation , 1986.

32. Engeles W.A.H., Zeelenberg C., *"A Single Scan Algorithm for QRS Detection and Feature Extraction ,"* IEEE Comput. Card., Long Beach: IEEE Computer Society ,1979.

33. Merri M. et al, *"Sampling Frequency of the Electrocardiogram for Spectral Analysis of the Heart Rate Variability,"* IEEE Trans. Biomed. Eng., vol 37, No. 1, pp.99-105, Jan 1990.

34. Murthy I.S.N., Ranjarai M.R. *"New Concepts for PVC Detection ,"* IEEE Trans. Biomed. Eng. vol. BME-26, No. 7, July 1979.

35. Oppenheim, Schafer, *"Digital Signal Processing,"* Prantice Hall International 1975.

36. Pinciroli et al, *"Changes in Health Care instrumentation due to Microprocessor Technology,"* North Holland Publishing Company 1980.

37. Rabiner, Gold, *"Theory and Application of Digital Signal Processing,"* Prantice Hall International 1976.

38. Stout D.F., Kaufman M., *"Handbook of operational Amplifiers and Circuit Design ,"* N.Y.: McGraw Hills, 1976.

INDEX